DL20 GROUP PROJECT - COMPARISON OF FEDERATED (DISTRIBUTED) VS. CENTRAL DEEP LEARNING FOR IMAGE CLASSIFICATION

Islamiia Gilmanova
islamiia.gilmanova@tu-ilmenau.de

Oleg Laptev
oleg.laptev@tu-ilmenau.de

Shubhesh Swain
shubhesh.swain@tu-ilmenau.de

Muhammad Ehtasham Ul Haq Janjua
muhammad-ehtasham-ul-haq.janjua@tu-ilmenau.de

Project Mentor: Daniel Scheliga
daniel.scheliga@tu-ilmenau.de

January 25, 2021

ABSTRACT

Artificial intelligence has made tremendous advancements in the past decade. This has become successful due to the high-speed computing device, growing AI community, and availability of the dataset. However, this has also become a growing concern for the privacy of the data when it is used for training the Machine learning models. This is where Federated Learning comes in handy. Federated Learning is the decentralized training technique where the Global model from the centralized server is shared across all its connected clients. These clients/agents train locally on their device using their own data and then share its weight to the central server which aggregates the weights. In this paper, we are describing the Federated Learning approach and compare it with the Centralized learning method by experimenting on the Intel Image Classification dataset [1].

1 Introduction

Over the past few years, data privacy has become a major concern among people as well as the humongous data growing day by day. Data such as hospital patient records, defense, telecommunication, etc are very sensitive, and sharing this data with a company for model training can lead to a data breach. In past few years there has been many incidents of data leak such as the famous Cambridge Analytica incident where Facebook shared its User data to the company. Many government has been very strict about this kind of incident and brought new laws to protect its Citizen's data. One good example of such law is the European Union GDPR and HIPPA in USA [2]. The companies working with the data which are very sensitive must adhere to this kind of laws to operate. Due to this Researchers are vastly focusing on how to maintain data privacy. The second biggest problem is size of the data is growing day by day. Data are gathered from many sources such as IoT device, smartphones, smartwatches etc. Training on such data requires a powerful hardware. All this problem has has led the Researchers to come up with the Federated learning algorithm which can train the AI/ML model on the dataset while maintaining the data privacy. Federated Learning was first termed by the Google in the year 2017. Since its first introduction, the topic has been adopted quickly by many AI communities for their research work which needs data privacy.

In a Federated Learning scenario, a central server shares the Global model across its permitted clients [3]. These clients gets the model weights and training program from the server. The client devices can be mobile devices, IoT devices, tablet devices, etc. The input data can be a image data or a text data which is specific to every client device. Once this client trains the model locally, it then shares the weight back to the central server. Once the central server receives all the weights, it then aggregates all the weights that it received. After the aggregating, the central server generates

the new Global model with the updated weights. Once we have this new model, the next round of training starts with the Global model being shared with the clients. This training procedure keeps running by the server-side until the training procedure is stopped by the person monitoring the process. The figure 1 shows the typical Federated Learning architecture.

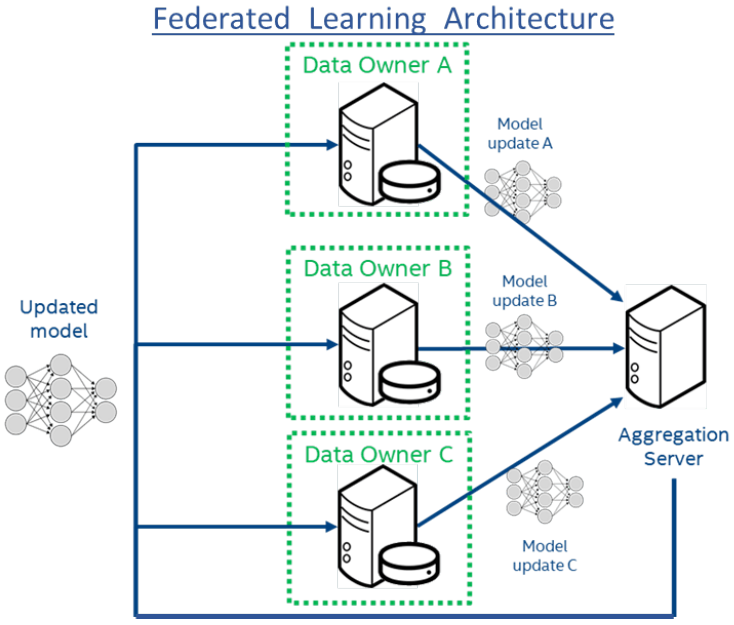

Figure 1: [4] Federated Learning Architecture

The motivation of this project is to compare between the Centralized and Federated Learning approach. For our use case, the input data for the Centralized and Federated Learning algorithm is the Intel Image Classification dataset and we are classifying the images. We have used the CNN architecture for both the case. The rest of the paper is divided as follows: Section 2 discusses about the related work, Section 3 tells about the dataset and the Features used in the experiment, Section 4 describes about the Methods used, Section 5 gives the detailed information about the design of the experiment and finally the Section 6 describes the Evaluation and Results and the report ends with the Conclusion and Future work in Section 7.

2 Related Work

In several research papers, Federated learning has been proposed as a setting alternative to the familiar learning: a shared global model is trained under the coordination of a central server, from a federation of participating device. H. Brendan McMahan with colleagues from Google AI introduced the "Communication-Efficient Learning of Deep Networks from Decentralized Data" in 2017 [5] where they introduced the Federated Averaging method(FedAvg) for the FL on distributed mobile devices. Communication costs and optimization algorithm showed up to 100 times better results than usual SGD.

In 2018 the article "Federated Learning for Mobile Keyword Prediction" was introduced by the Google LLC team (A. Hard et al) [6] where the FedAvg method was implemented for learning of the recommendation neural engine of Google Keyboard application. Then, in 2019 McMahan's team and specialists from top IT universities introduced the "Advances and Open Problems in Federated Learning" book [3] where the overall review on the Federated approach was given. Finally, in 2019 paper "Towards federated learning at scale: System design" [7] was released by Bonawitz K. et al summarizing all the most effective methods of federated learning in high-level system design, observing challenges and touching upon the open problems and future work in the field.

Going back, let's describe the `FederatedAveraging` algorithm which will be used in the training and evaluation process of our approach in further chapters. The algorithm that was proposed in 2017 by H. B. McMahan [5] describes the method of averaging the model and the learning for the Federated API. In the algorithm there is a typical implementation of `FedSGD` with $C = 1$, where C - fraction of clients on each round that controls the global batch size,

so *C = 1* corresponding to full-batch (non-stochastic) gradient descent, and a fixed learning rate η has each client *k* compute $g_k = \nabla F_k(w_t)$, the average gradient on its local data at the current model w_t, and the central server aggregates these gradients and applies the update $w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$, since $\sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t)$. An equivalent update is given by $\forall k$, $w_{t+1}^k \leftarrow w_t - \eta g_k$ and then $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$. That is, each client locally takes one step of gradient descent on the current model using its local data, and the server then takes a weighted average of the resulting models. Once the algorithm is written this way, we can add more computation to each client by iterating the local update $w^k \leftarrow w^k - \eta \nabla F_k(w^k)$ multiple times before the averaging step. We term this approach `FederatedAveraging` (or `FedAvg`). The amount of computation is controlled by three key features: *C*, the fraction of clients that perform computation on each round; *E*, the number of training passes each client makes over its local dataset on each round; and B, the local minibatch size used for the client updates. We write $B = \infty$ to indicate that the full local dataset is treated as a single minibatch. Therefore, at one endpoint of this algorithm family, we can take $B = \infty$ and *E* = 1 which corresponds exactly to `FedSGD`. For a client with η_k local examples, the number of local updates per round is given by $u_k = E\frac{n_k}{B}$; The pseudo-code realization of the `FedAvg` is given in Algorithm 1 [5].

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and is the learning rate.

Server executes:
initialize w_0
for *each round t = 1,2,...* **do**
  m←max(C • K,1)
  $S_t \leftarrow$ (random set of m clients)
  for *each client* $k \in S_t$ ***in parallel*** **do**
    $w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
  end
end
ClientUpdate(*k, w*): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size *B*)
for *each local epoch i from 1 to E* **do**
  for *batch* $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla l(w; b)$
  end
end
return w to server

3 Dataset and Features

The input of the approach is the Intel Image Classification Dataset containing image data of Natural Scenes around the world [3]. The overall data of 25k images are divided into three packages:

• seg-pred – unspecified 7 thousands of 150x150 pixels images of different classes;

• seg-train – 14 thousand of 150x150 pixels images classified in 6 categories;

• seg-test – 3 thousand of 150x150 pixels images classified in 6 categories.

These six categories of places include : ('buildings' -> 0, 'forest' -> 1, 'glacier' -> 2, 'mountain' -> 3, 'sea' -> 4, 'street' -> 5). Each of the segments is separated in a ZIP compressed file.

The *seg-train* and *seg-test* sets are to be used in both are approaches - centralized and federated to train the model end evaluate the results of training. However, the *seg-pred* unlabeled dataset can be used only for visualization of the evaluation of classification, so it is not that important, but might be used for more visual results. Both approaches will work with 150x150 3-color-dimensional image data.

4 Methods

The research project is divided into two main parts: first – the implementation and training of the CNN employing centralized learning with the evaluation of performance, the second is the implementation of the federated learning

approach and evaluation of the same network architecture trained in a federated way. The Convolutional network can be implemented layer-wise by the Keras framework. The Federated learning approach is to be implemented by the tools of TensorFlow Federated API to simulate the device node communications and learning process on the distributed devices.

For the first part, the dataset has to be preprocessed, and the implemented CNN architecture should be trained on it, where for evaluation performance accuracy of the classification and learning time will be used. The second part includes the creation of the TFF(TensorFlow Federated) model based on the same CNN architecture to simulate the distributed learning on user's devices.

5 Design of Experiment

5.1 Centralized based Image Classification

Starting with the acquisition of data, we made use of *kaggle* API to fetch Intel Image Classification zipped data on Colab and later extracting it in a newly made directory for dataset. When it comes to Image classification in deep learning CNN's have been proved as the most accurate one. Before discussing the CNN architecture it is pertinent to mention about the data preprocessing needed before feeding the input to convolution neural network.

A method is defined to first order images with respect to their respective labels and then returning a shuffled dataset of images with corresponding labels. It is followed the image conversion to arrays and displaying of some samples with corresponding labels using the matplotlib library.

5.1.1 Image Data Generator Class

We have used the keras offered image data generator class. It lets to augment the images in real time while the model is still training to make model more robust and saving overhead memory. Another advantage of augmentation is to increase the diversity of data available.This helps in improving the generalize ability of a model.

5.1.2 CNN Architecture

We have used Sequential API of Keras models to create our CNN architecture. It consists of 2D convolution layers with appropriate activation functions followed by Maxpooling layers to reduce the spatial dimensions of feature map. A flatten layer to convert into single dimensional vector which is later processed into Dense layers with Softmax activation function at last layer. The architecture is shown in Figure 1.

5.1.3 Optimizers, Loss functions, Accuracy and Training

During compilation of model SGD optimizer was used with $lr = 0.001$ and since its multi-class classification problem therefore categorical cross entropy loss is used as loss function. A simple accuracy measure is also added. We trained the classifier for 27 epochs.

5.2 Federated baseline

Starting with the acquisition of data, we again used *kaggle* API to fetch Intel Image Classification into Colab. Then image data was pre-processed to satisfy TFF requirements for data to being in the tf.data.Dataset format. Then, the population of clients were made and the set was distributed between them.

5.2.1 Preprocessing of Dataset for TensorFlow Federated API

For preprocessing the images the tf.keras.preprocessing.image-dataset-from-directory library was used. It extracted the images from the packages and then shuffled it. Unfortunately the data here is not augmented much because of specific input requirements of TFF and absence of possibility to augment in this library. The data was distributed with minibatch size of 32 and then given to tff.simulation.ClientData function.The test data were made in the same manner.

5.2.2 CNN baseline Architecture

The architecture of the model is the same as in the centralized approach. It has one input layer, three Conv2D layers with max-poolings with number of filters 16,32 and 64, one flatten layer, dense layer with ReLU activation and the final dense. The final layer here is without Softmax to satisfy the TFF requirements for federated Sparse-Categorical Cross-Entropy. The federated optimizer here is the federated Adam.

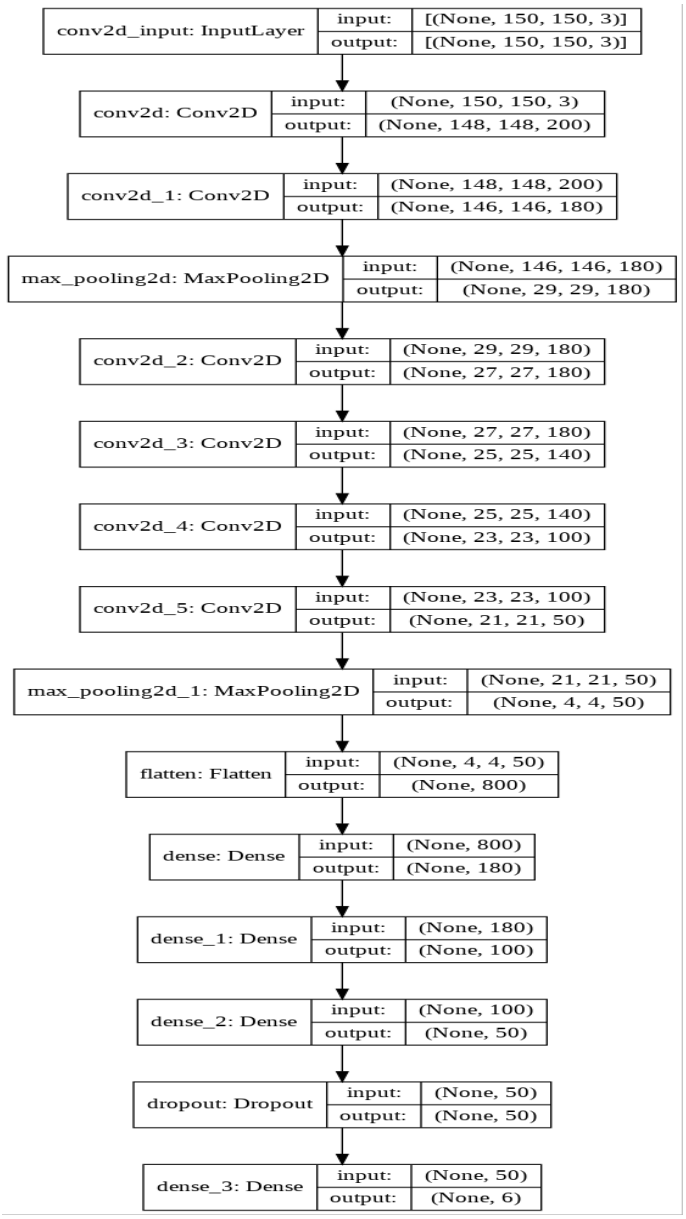

Figure 2: CNN architecture

5.2.3 Simulating Distributed Devices

After the pre-processing and the definition of CNN we were creating the simulated client devices, for the first time only in range of 10. We distributed the training dataset randomly and saved the sample. With the method build-federated-averaging-process we compiled the model with optimizers and then, after initialization and defining the number of rounds start the `FedAvg` training. The number of rounds which we had time to implement and test was only 13 because of the very slow Federated Averaging algorithm and limited hardware resources. On the Fig.3 presented the 8 of 10 clients with train data distribution. We can see that privacy of the data was saved because of the complete random distribution of data.

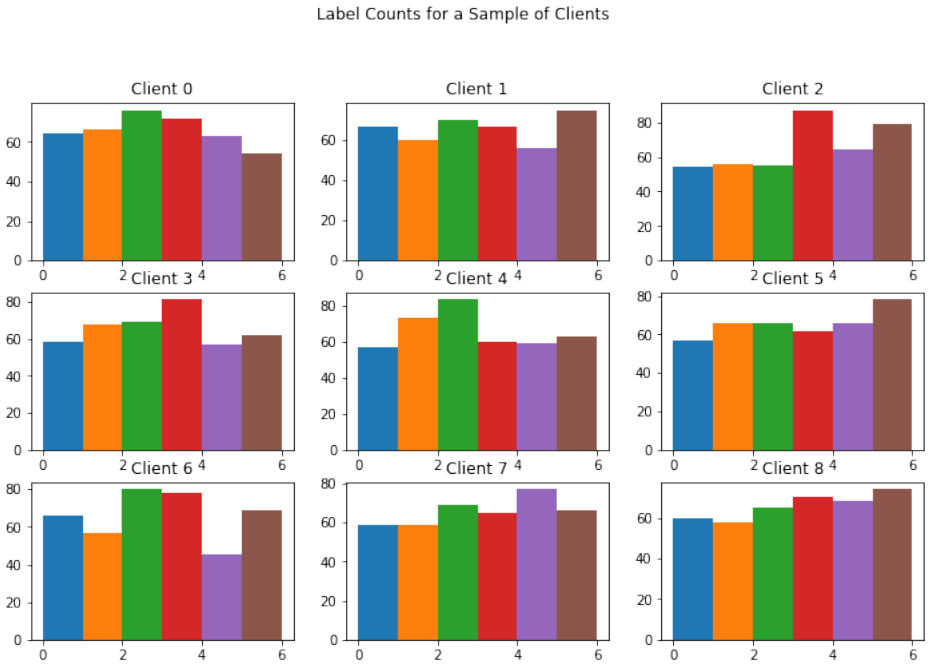

Figure 3: Client data distribution

6 Evaluation and Results

We have plotted the curves for training and validation accuracies and losses. The centralized approach training ended with 0.3027 accuracy and loss = 1.74 . The training curves corresponding to the learning process are in the Fig.4.

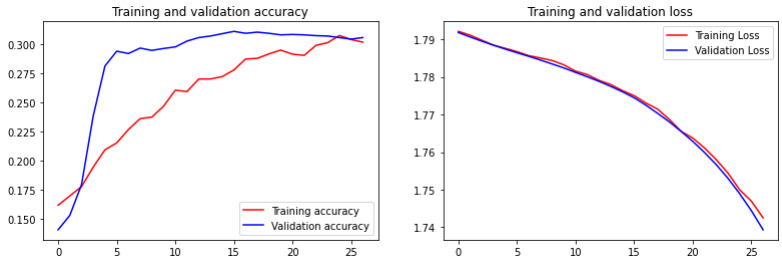

Figure 4: Loss curves

At last predicted for the results on prediction directory images and plotted the images with prediction score using Matplotlib.The results of prediction can be seen in Fig.5.

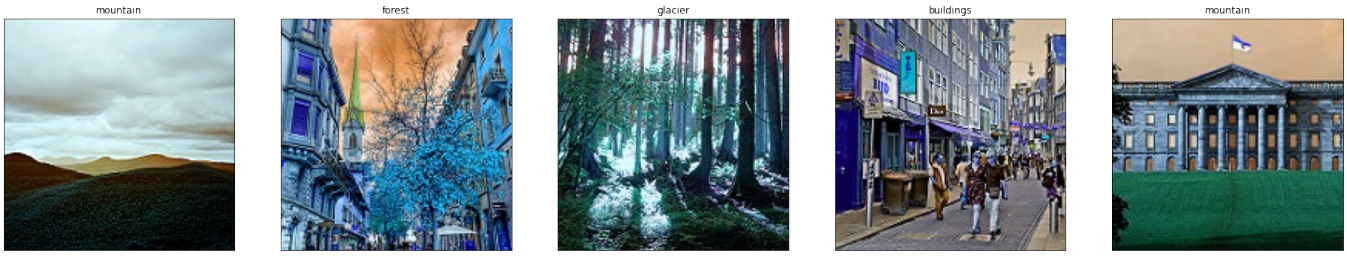

Figure 5: Predicted labels from centralized model on seg-pred

Model	Accuracy	Loss
Centralized Approach	0.3027	1.74
Federated approach	0.1855	8.02

Table 1: The Performance comparison for the Federated and Centralized approaches

For that little time - 3 hours of learning we have got 18% accuracy and loss=8.02 results on test set.The training curves were plotted by Matplotlib. The results for the federated approach are presented in Fig.6 and Fig.7.

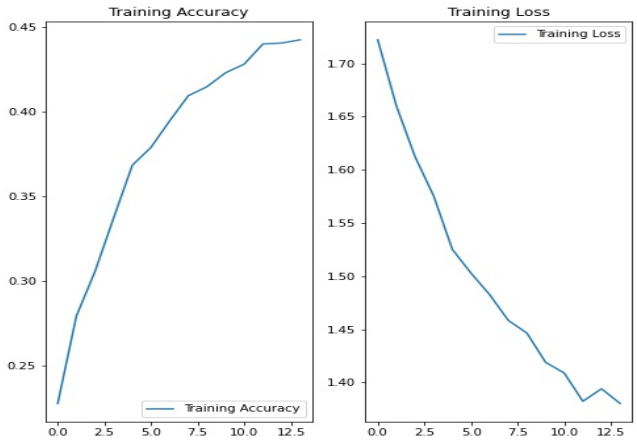

Figure 6: Accuracy and loss curves

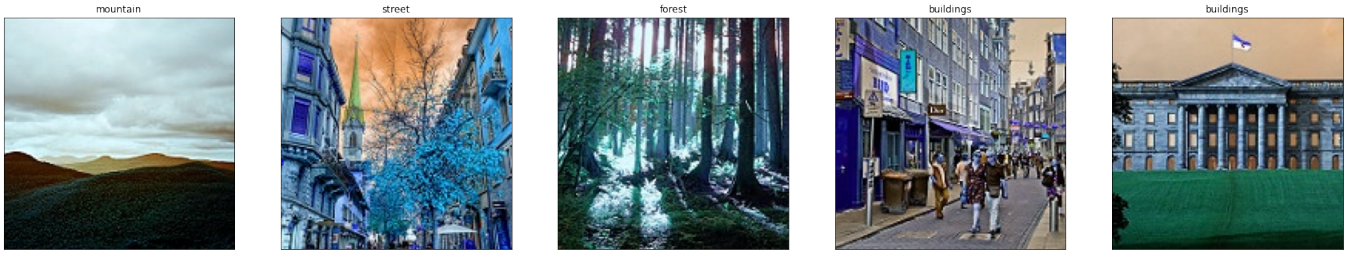

Figure 7: Predicted labels from the Federated approach

Evaluating the results of the work that was made we can say that the Accuracy-Loss performance was quite standard for both Centralized and Federated approaches, in case we removed the augmentation section from the Centralized one. It was made to make both the models comparable. It can be seen at the Table 1.

Another metric we used was the prediction accuracy by both of the approaches. For this case Federated and Centralized models were converted to HDF5 format and then both of them were run on the same set of the 5 randomly given images from seg-pred dataset. Here we found the surprising results. The predictions were not so good, but the Federated model showed the better accuracy on the picture data.

7 Conclusion and Future Work

By making the research deep into the simulations of data we understood that the Federated approach really makes the data private by randomly putting the different amounts of images from each of the classes to the clients. Moreover, we did the successful comparison of the approaches performances that is shown in the Table 1. And a comparison of predictions on the test 5-image batch. The source code of the project is uploaded to the special repository in GitHub [8].

In the future, it can be beneficial if we increase the number of epochs over 200 and augment the dataset which will make the model more robust. The same approach can tested on various kinds of dataset. In addition, we can test the accuracy of the model by increasing the number of client devices.

8 Contributions

This research project consists of four team members. The work of Islamiia Gilmanova, Oleg Laptev, and Shubhesh Swain is dedicated to the implementation, testing, and evaluation of the Federated approach – Ms. Gilmanova is simulating distributed devices, and Mr. Laptev is implementing the neural network architecture, Mr. Swain is implementing server simulations and doing the reports on the work. The responsibilities of Mr. Janjua are focused on the centralized approach for the classification task – he implements the conventional image classification algorithms in Keras framework of TensorFlow and evaluating the results.

References

[1] Puneet Bansal. *Intel Image Multiclass Scenes Classification Dataset*. © Original Authors/ Intel, Gurugram, Haryana, India, 2018. `https://www.kaggle.com/puneet6060/intel-image-classification`.

[2] Olivia Solon. Facebook says cambridge analytica may have gained 37m more users' data (2018). *URL https://www. theguardian. com/technology/2018/apr/04/facebook-cambridge-analytica-user-data-latest-more-than-thought*, 2018.

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

[4] Brandon Edwards Jason Martin Spyridon Bakas Tony Reina, Micah J Shelle. Federated learning for medical imaging (2018). *URL https://www.intel.com/content/www/us/en/artificial-intelligence/posts/federated-learning-for-medical-imaging.html*.

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

[6] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. *Federated learning for mobile keyboard prediction*. 2018.

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

[8] Islamiia Gilmanova, Shubhesh Swain, Muhammad Ehtasham UI Haq Janjua, and Oleg Laptev. Comparison of federated (distributed) vs. central deep learning for image classification. *https://github.com/wavatatarskii/COMPARISON-OF-FEDERATED-DISTRIBUTED-VS.-CENTRAL-DEEP-LEARNING-FOR-IMAGE-CLASSIFICATION*.