Building Decentralized Image Classifiers with Federated Learning

Judy T. Raj

Backend, Orcablue.ai
Bangalore, India, 560102

judytraj007@gmail.com

***Abstract*— The commercial use of neural networks has been greatly curbed by data privacy concerns. As long as the accumulation and use of private data is regarded necessary for integrating neural networks into products, consumers will be reluctant to use or allow access to any deep learning integrated product and producers will be equally deterred from leveraging deep learning for performance improvement. Federated learning was first introduced as a solution to this conundrum in a 2016 paper published by Google titled Communication-Efficient Learning of Deep Networks from Decentralized Data[1]. In this study, we examine how the performance of a decentralized image classifier compares to that of a centralized one. The performance of an image classifier trained across ten devices was compared to a model built with the same architecture but trained centrally on one corpus of training data. The outcome demonstrates that the decentralized model compares quite well to the centrally trained classifier in terms of accuracy, precision and recall.**

***Keywords*— convolutional neural network, federated learning, distributed computing, decentralization, image classification, computer vision**

I. Introduction

As machine learning and data driven business models continue to rise in popularity, data privacy concerns have become prevalent. Neural networks cannot achieve their full business potential unless they can be integrated into products without causing privacy concerns for the consumers.

Federated learning[1] is a technique of training machine learning models in which the model is copied to and trained on different machines and the weights from each of the trained models are aggregated to update the models. The technique as introduced by Google in the Google keyboard[2] proceeds as follows. The model architecture is defined and is trained on some open sourced dataset. The model is downloaded to user machines where it is trained on the locally available data. The resultant weights are uploaded to a central server which aggregates the weights sent from all the machines. These optimum weights are used to update the original model. The machines involved in the training syncs with the server later to update the local version. Thus the models across this network of machines become trained and updated with the full power of data spread across a cluster of machines without the data ever having left the original storage devices.

No training data is sent from the devices to the server. This approach allows products to leverage the full might of neural networks for performance optimization and personalisation while being able to guarantee complete data privacy.

In this paper, we discuss a convolutional network[3] for binary classification[4] trained on a centralized corpus of data and compare its performance with a classifier of the same architecture trained using the federated learning approach.

II. ARCHITECTURE OVERVIEW

This section provides an overview of the model architecture, the dataset used for training, the target output, and the data preprocessing and augmentation techniques used. The model is built and trained using the PyTorch Deep Learning Framework[5] and is based on the version of alexnet provided by the torchvision module in PyTorch.

978-1-7281-7366-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Temple University. Downloaded on August 10,2021 at 21:31:30 UTC from IEEE Xplore. Restrictions apply.

A. Data Preprocessing

The model was based on the Kaggle cats and dogs dataset[6] and was built to perform binary classification. The dataset contains 25,000 labelled RGB images of dogs and cats. We used the image transformations available in the torchvision.transform[5] module to preprocess the images and augment the dataset. The dataset consisted of images of varying sizes. They were all resized to be 224 x 224 and were converted to tensors using the respective methods. The images were additionally submitted to random horizontal flips and center cropping and normalized using the mean and standard deviation values [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225] respectively.

The mean and standard deviation values were obtained from the ImageNet dataset[7]. The architecture for the models are based on the pretrained model alexnet[8], built on the ImageNet dataset, as we employ transfer learning[9] to perform the classification. Transfer learning[10] is a technique of training a neural network where instead of defining a new model architecture and initialising it with random weights, the architecture or weights from a model trained for a similar purpose is copied. The batch size was set to four for both the models. Fig 1. shows one such batch of data.

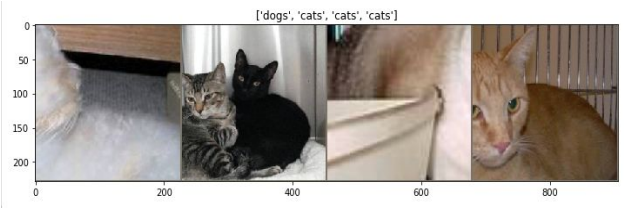


Fig. 1 One batch of training data and labels

B. Model Architecture

The models subpackage in the torchvision package of PyTorch provides various model architectures as well as pretrained weights for image classification. Alexnet was the convolutional neural network that won the 2012 ImageNet Large Scale Visual Recognition Challenge[11] with a top-5 error of 15.3%. The model expects input images in mini-batches of 3-channel RGB images of shape 3 x H x W, where H and W are expected to be at least 224. The images have to be loaded into a range of [0, 1] and then normalized using the mean and standard deviation. We used the alexnet implementation provided by torchvision[4] to define our model's architecture and initialized the model with the pretrained weights. The last linear layer is redefined for binary classification by setting the number of output features to 2. Fig. 2 shows the final architecture[12] of the binary classifier.

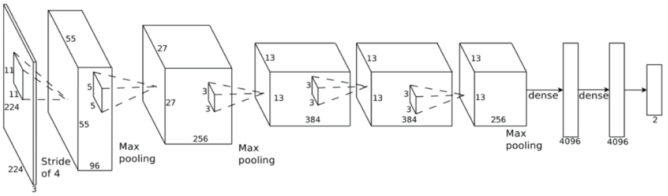


Fig. 2 Architecture of alexnet for binary classification

III. CENTRALIZED CLASSIFIER

This section discusses training the centralized classifier. The model was initialised with pretrained weights obtained from being trained on ImageNet.

A. Training

The image corpus was split into training and test sets. The training set included 999 images of cats and 1004 images of dogs. The test set contained 506 images of cats and 507 images of dogs. The cross entropy loss[13] was used as the loss function[15] and the Stochastic Gradient Descent algorithm[14] was used for optimization.

The training data was fed in batches of four. The model was trained over 10 epochs with a learning rate of 0.001[16]. On a single CPU with 8GB RAM and 1.8 GHz Intel Core i5, the training process took approximately 120 minutes. Since our dataset is very similar to the images contained in the dataset on which alexnet was trained on, the training process here only requires finetuning the pretrained weights to fit the new data[17]. This scenario will often be true in actual use cases of federated training as the models will be first defined and trained on some available data before being

978-1-7281-7366-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Temple University. Downloaded on August 10,2021 at 21:31:30 UTC from IEEE Xplore. Restrictions apply.

integrated into some product that is sent to multiple user devices for further training[18].

Fig. 3 shows the values of the loss during the training[18]. The final value of loss after the ten epochs was 0.01521.

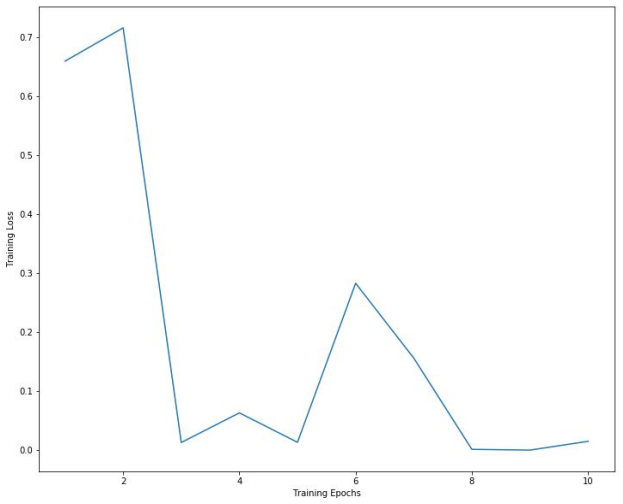


Fig. 3 Loss convergence status of centralized classifier

B. Results

The final model was tested on 506 images of cats and 507 images of dogs. Transforms were applied to the images to resize them to 224 x 224 and convert to tensors. The model showed an accuracy score of 0.9417, a precision score of 0.9627 and a recall score of 0.91897. The values were computed using the methods from the sklearn metrics library[19]. Table 1 shows the confusion matrix[20] for the centralized classifier.

TABLE I
CONFUSION MATRIX FOR CENTRALIZED CLASSIFIER

	0	**1**
y_true	488	41
y_pred	18	465

IV. DECENTRALIZED CLASSIFIER

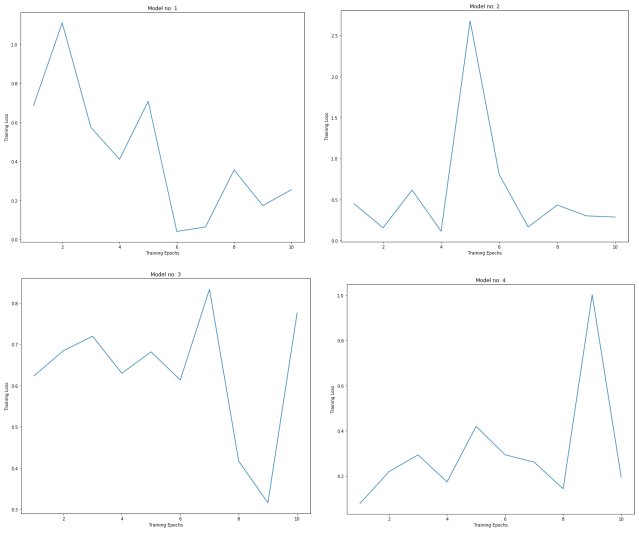
This section provides an overview of the steps followed in building the decentralized classifier. We used the same alexnet model as the one used to define the centralized model and initialized it with the same pretrained weights from the torchvision package.

A. Training

The same Kaggle cats and dogs dataset was taken to train the decentralized model. The dataset was split into ten subsets, each of which contained 99 images of cats and 99 images of dogs. All the images were resized, randomly flipped, normalized and converted to tensors by applying the same torchvision transforms as before.

B. Federated Learning

A model object was defined with the same alexnet model architecture as shown in Fig 2. and was initialised with the same pretrained weights as the centralized classifier. The model was then duplicated to make a total of ten models. All ten models were trained on one of the ten distinct subsets of the image dataset. The models were trained for ten epochs each which involved fine tuning the pretrained layers to fit to the new data. The training data was fed in batches of four as shown in Fig 2. The cross entropy loss function[21] and the Stochastic Gradient Descent optimizer, as provided by PyTorch were applied to each of the ten models. Each model took approximately 20 minutes to finish training. Fig. 3 shows the training loss for the sub models.



978-1-7281-7366-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Temple University. Downloaded on August 10,2021 at 21:31:30 UTC from IEEE Xplore. Restrictions apply.

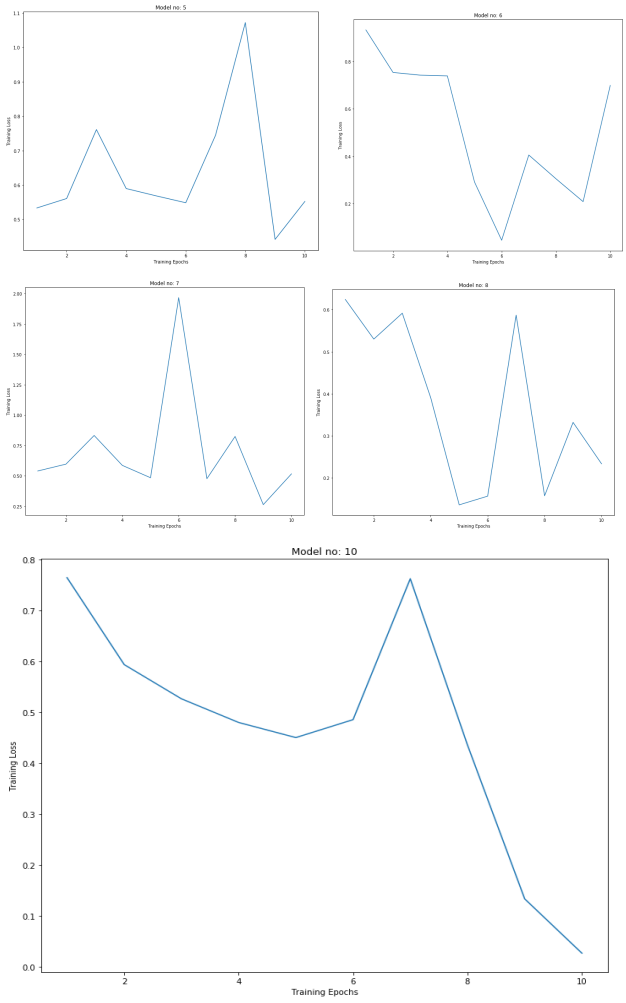


Fig. 3 Loss convergence status of the sub models

Each of the submodels were tested on the same dataset we tested the centralized classifier on. The table II shows the accuracy, precision and recall scores of each of the ten submodels tested on the same images. The performance metrics were evaluated using the sklearn metrics library. The test set contained 506 images of cats and 507 images of dogs.

TABLE II
PERFORMANCE METRICS FOR SUBMODELS

Model no.	Accuracy	Precision	Recall
1	0.8202	0.8785	0.7431
2	0.8804	0.8431	0.9348
3	0.8349	0.9007	0.7529
4	0.8083	0.7644	0.8913
5	0.6679	0.6986	0.5909
6	0.8488	0.8966	0.7885
7	0.8122	0.8264	0.7905
8	0.81027	0.7793	0.8656
9	0.8557	0.8203	0.9111
10	0.8636	0.9107	0.8063

C. *Model Aggregation*

Once all the models were trained and tested, we aggregated the weights using the polyak averaging[22] method. Polyak averaging consists of averaging several points in the parameter space that the optimization algorithm traverses through. The weights were computed using the equation weights_new = k * weights_old + (1 - k) * weights_new and stored in a dict. The value 0.5 was assumed for k. We next defined a new object of the alexnet model with the same architecture[23] as Fig 2.

The model was not assigned with the pretrained weights and just the architecture was downloaded from torchvision. The code snippet below shows the model aggregation steps:

```
beta = 0.5
params = models_[0].named_parameters()
dict_params = dict(params)

for i in range(1, num_models):
  for name, param in    models_[i].named_parameters():
    if name in dict_params:
        dict_params[name].data.copy_(beta * param.data +
                    (1 - beta) * dict_params[name].data)
```

Pytorch provides the load_state_dict() method to load saved weights to a model object. We used the load_state_dict() method to load the newly aggregated weights to the newly defined model.

D. *Results*

The final model after being loaded with the aggregated weights of the ten submodels were tested on the original test set which contained 506 images of cats and 507 images of dogs. The decentralized model showed performance metrics that were at par with that of the centralized model.

978-1-7281-7366-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Temple University. Downloaded on August 10,2021 at 21:31:30 UTC from IEEE Xplore. Restrictions apply.

The decentralized model showed an accuracy score of 0.9348, a precision score of 0.9198 and a recall score of 0.9526. The values were computed using the same methods from the sklearn metrics library which we used to evaluate the centralized classifier. Table III shows the confusion matrix for the decentralized model.

TABLE III
CONFUSION MATRIX FOR DECENTRALIZED CLASSIFIER

	0	**1**
y_true	464	24
y_pred	42	482

IV. Conclusions

Federated learning is a viable alternative to the traditional approach of building predictive models from a centralized corpus of data. Locally trained weights can be used to replace user data being harvested from individual devices. Collecting locally trained weights instead of training data from devices is a safer alternative to the traditional centralized models. Federated learning at scale[24] has the potential to quell the data privacy concerns and lead to popularizing the use of neural networks in products for performance improvement and personalisation. With federated learning ensuring data privacy[25], neural networks can be expanded to their fullest commercial potential.

Our decentralized model performed slightly better than the centralized model when tested on the same data. With the guarantee of zero data harvesting, the vast network of mobile phones and laptops with high computational powers and immense amount of data becomes fully accessible to federated learning. With access to larger amounts of data than can be provided in a central corpus, federated learning has the potential to build much better performing models than the ones built using the traditional centralized approach. In conclusion, decentralized models offer the same performance whilst ensuring data privacy and requiring less dedicated computation resources, thus making a strong case to be the future of deep learning.

References

[1] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas, Communication-Efficient Learning of Deep Networks from Decentralized Data, Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017. JMLR: W&CP volume 54, arXiv:1602.05629 [cs.LG]

[2] [online] Available: http://cs231n.github.io/convolutional-networks/.

[3] Brendan McMahan and Daniel Ramage, Federated Learning: Collaborative Machine Learning without Centralized Training Data, [online]Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[4] B Liu, Y Liu, K Zhou, Image classification for dogs and cats, [online] Available: https://sites.ualberta.ca/~bang3/files/DogCat_report.pdf.

[5] [online] Available: https://pytorch.org/docs/stable/index.html.

[6] Dogs vs Cats, [online] Available: https://www.kaggle.com/c/dogs-vs-cats.

[7] A. Krizhevsky, I. Sutskever, G. E. Hinton, "Imagenet classification with deep convolutional neural networks", Advances in neural information processing systems, pp. 1097-1105, 2012.

[8] Alex Krizhevsky, One weird trick for parallelizing convolutional neural networks, [online] Available: https://arxiv.org/pdf/1404.5997.pdf.

[9] Sasank Chilamkurthy, TRANSFER LEARNING FOR COMPUTER VISION TUTORIAL, PyTorch Tutorials, Accessed on: January 13, 2020. [Online]. Available*: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html*

[10] Pan, S.J. and Q. Yang, A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 2010. 22(10): p. 1345-1359.

[11] Deng, J., et al. Imagenet: A large-scale hierarchical image database. in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. 2009. IEEE.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, vol. 5, no. 2, pp. 1929-1958, Jun 2014.

[13] Zhilu Zhang, Mert R. Sabuncu, Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels, arXiv:1805.07836v4 [cs.LG] 29 Nov 2018

[14] Sebastian Ruder, An overview of gradient descent optimization algorithms, arXiv:1609.04747v2 [cs.LG] 15 Jun 2017

[15] Katarzyna Janocha, Wojciech Marian Czarnecki, On Loss Functions for Deep Neural Networks in Classification, arXiv:1702.05659v1 [cs.LG] 18 Feb 2017

[16] Xiaofei Zhang, Yi Zhang, Erik Y. Han, Nathan Jacobs, Qiong Han, Xiaoqin Wang, Jinze Liu, *Classification of whole mammogram and tomosynthesis images using deep convolutional neural networks,* DOI 10.1109/TNB.2018.2845103, IEEE Transactions on NanoBioscience

[17] KIEN NGUYEN, CLINTON FOOKES, ARUN ROSS AND SRIDHA SRIDHARAN, Iris Recognition With Off-the-Shelf CNN Features: A Deep Learning Perspective, Digital Object Identifier 10.1109/ACCESS.2017.2784352 [online] Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8219390

[18] Qiang Yang, Yang Liu , Tianjian Chen, Yongxin Tong, Federated Machine Learning: Concept and Applications, ACM Transactions on Intelligent Systems and Technology (TIST)January 2019 Article No.: 12 https://doi.org/10.1145/3298981

[19] A. Howard, "Some improvements on deep convolutional neural network based image classification", ICLR, 2014.

[20] J. Redmon, A. Angelova, "Real-time grasp detection using convolutional neural networks", IEEE International Conference on Robotics and Automation, pp. 1316-1322, 2015.

[21] K. Yanai, Y. Kawano, "Food image recognition using deep convolutional network with pre-training and fine-tuning", 2015 IEEE

978-1-7281-7366-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Temple University. Downloaded on August 10,2021 at 21:31:30 UTC from IEEE Xplore. Restrictions apply.

International Conference on Multimedia & Expo Workshops (ICMEW), pp. 1-6, 2015.

[22] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, Andrew Gordon Wilson, *Averaging Weights Leads to Wider Optima and Better Generalization*, arXiv:1803.05407v3 [cs.LG] 25 Feb 2019.

[23] Hang Chang, Cheng Zhong, Ju Han, Jian-Hua Mao, "Unsupervised Transfer Learning via Multi-Scale Convolutional Sparse Coding for Biomedical Application", IEEE Transactions on Pattern Analysis and Machine Intelligence, janvier 2017.

[24] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecn, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, Jason Roselander, TOWARDS FEDERATED LEARNING AT SCALE: SYSTEM DESIGN, [online] Available: https://arxiv.org/pdf/1902.01046.pdf

[25] R. Shokri, V. Shmatikov, "Privacy-preserving deep learning", Proc. 22nd ACM SIGSAC Conf. Computer and Communications Security, pp. 1310-1321, 2015.

978-1-7281-7366-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Temple University. Downloaded on August 10,2021 at 21:31:30 UTC from IEEE Xplore. Restrictions apply.