Evaluating the Efficacy of Federated Learning in Comparison to Centralized Models Using Two Distinct Image Classification Architectures

Siddharth Balaji
Homestead High School
August 25, 2024

Abstract

This experiment explores the viability of federated learning when used with two image classification AI models for melanoma. Federated learning, a machine learning strategy to keep data local to certain clients to ensure data privacy, is mainly used for work between multiple corporations (hospitals, large companies, etc.) and is necessary in order to maintain data privacy. For hospitals in particular, the HIPAA law prohibits the distribution of patient data without patient consent [4]. The results of the centralized model and the federated learning model are compared to show whether this new approach to data privacy can truly be trusted in this day and age. Two different image classification models, a basic CNN (convolutional neural network) and a more advanced CNN, an AlexNet, will be used in this test for comparison. The Flower framework will be used in order to implement federated learning with the two models. The results of this experiment show that models implemented with federated learning are still comparable in accuracy to the centralized models. This experiment is important because it tests whether patient data can be kept private while also making sure that the data can be used for AI training.

Introduction

Melanoma is a type of skin cancer that is caused by a tumor composed of melanocytes, specific cells that produce pigments in the human body. This type of cancer is very rare, affecting around 22.1 out of 100,000 people in 2019. It is typically caused by high sun exposure to fair-skinned people. It was first recorded in western literature in 1651 by Drs. Highmore and Bonet [1].

Although melanoma is considered a serious form of cancer, it is treatable through modern medical technology. One of the most efficient forms of treatment for melanoma is immunotherapy. Immunotherapy uses the patients' own immune system to fight the cancer present, typically as a liquid that can enter the patients' veins or as a pill. The reason that immunotherapy is so effective for curing melanoma is because the tumor formed by melanoma is successful in creating antibodies to respond to the cancer. This strategy for melanoma in specific has four stages that all work together to successfully kill off the cancer [2, 3].

However, methods like this are only useful when the melanoma is detected early in its stages. If diagnosed too late in its stages, the chance of metastasis from the melanoma tumor increases significantly. For this reason, it is important that models exist that are able to detect melanoma early. While this model cannot completely replace doctors, it can be used to direct

patients to the hospital if the model is returning a positive test for melanoma. It can serve as a preliminary check before visiting and receiving treatment from doctors.

Image classification AI models can be extremely useful in classifying melanoma for a patient. These models typically have some of the data being used for “training” the model while the rest of the data is used for “testing” the accuracy of the now trained model. There are many types of image classification models, but all use this same form of training and testing the model. The conventional way to store this data is locally on the server where the model is trained and tested.

The main issue with this solution is that many hospitals want and are required to keep their patients’ data private. The Health Insurance Portability and Accountability Act (HIPAA) requires that patient data is kept confidential unless given permission from the patients. Due to this, many hospitals cannot contribute patient data [4]. Many of these AI models need a large dataset that contains real melanoma data in order to make as efficient of a model as possible. Since AI models typically store their datasets on a centralized server, hospitals do not want to send their own patients’ data to a potentially not trustworthy source where they do not know what will happen to their data.

Not only that, but many image classification models such as ones that detect melanoma may require data from more than one hospital. This is necessary to make the model as efficient and accurate as possible. Hospitals do not want to share patients’ data between each other similar to how they do not want to share it with a central server. This is the main issue of data privacy in relation to hospital cooperation.

In order to solve this problem of data privacy in AI models, federated learning was developed. In short, a model is created on a central server and the model is sent to each of the “clients”, in this case separate hospitals. The model is trained on each client server and sent back to the central server, where all the models are aggregated together to create one model. For example, a certain company creating a model may be considered as the central server while each hospital that is being requested for data can be considered a client server. This keeps data private while also making an efficient model.

This project aims to test the efficiency of federated learning on an image classification model. While federated learning does keep data private, it may not be as accurate as if the data was all on one server. Through tests, a model will be compared in accuracy when implemented with and without federated learning to see the avail of federated learning.

Background:

Federated Learning

As stated in the introduction, federated learning is a machine learning approach that decentralizes training and testing data. This approach is used in situations where it is required for data to be kept private to each client, typically when multiple companies or sources are working together by providing data for a model.

Not only is federated learning required in order to maintain data privacy, but there are also flaws in keeping all the data in one central server like most AI models do. For example, centralized data storage leads to potential threats to this data being leaked. One dataset is more vulnerable to hacks as opposed to the data being on multiple servers. Additionally, the

increased size of one central dataset leads to potential for the owner of the data to be leaked. Through data re-identification, the data can be matched with who the data belongs to [5].

In order to create a federated learning machine learning program, first a model needs to be created on a central server. This central server is typically where the creator creates their model. After this, individual devices, known as clients, are sent this model and update it based on their datasets. Once the model is trained on each of the client datasets, each of these new models are sent back to the central server. All the models are then aggregated to create a new and distinct model that is more accurate. This process keeps the data private while also reducing the load being placed on the central server [6]. However, there is another approach to federated learning. Instead of aggregating all the locally trained models, it is possible to go from client to client improving the model between each step. Once it has gone through all the clients, it will be then returned to the client server. This works so the model goes through all the clients before returning to the central server while the other approach sends the trained model from each client back to the client server, treating each client individually [7]. For the purposes of this experiment, this client to client approach will not be used, but instead the aggregation method will be used.

For real world experimentation, there would actually be separate datasets that are coming from different clients. However, for the purposes of this experiment, one dataset will be used and this dataset will be split among the separate clients. Only the training dataset will be split since that is what matters for training. This will be conducted by using a Federated dataset that has the dataset split up. The testing data will not need to be split up since the testing will be conducted on the central server once the models have been trained and aggregated.

Image classifiers

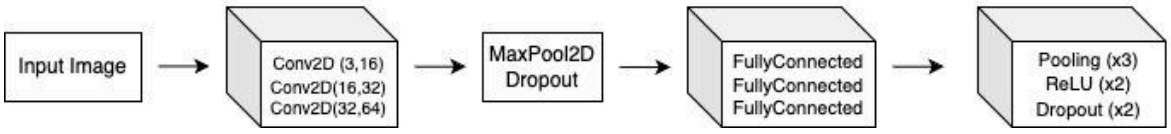
Image classifiers are a type of AI model that are able to identify images as part of certain “classes” of objects using specific algorithms. For the purpose of this paper, two separate, but similar image classifier AI models were used: a basic convolutional neural network (CNN) and a more advanced type of CNN, an AlexNet. While these two may be similar in structure, they both have different results for their implementation with federated learning and their accuracy when used as a centralized model.

To begin with, both of these models have the term “neural network” in it. A neural network is a form of machine learning in which the computer can do some task after learning from a “training” set of data. These networks are organized into layers of nodes which the data moves through one by one. Each of the nodes in these layers has a weight that determines whether the data in that node is passed onto the next node. Both CNNs and AlexNets are forms of neural networks [8].

There are many uses for image classification models in a wide range of fields. To begin with, they are frequently used in the medical field for cellular part identification and for identifying diseases, like this project is. For example, one experiment used another type of model, a DIU-Net in order to identify tumors from breast cancer. Image classifiers have been used for experiments such as this and for other image classification [13].

Convolutional Neural Networks (CNN)

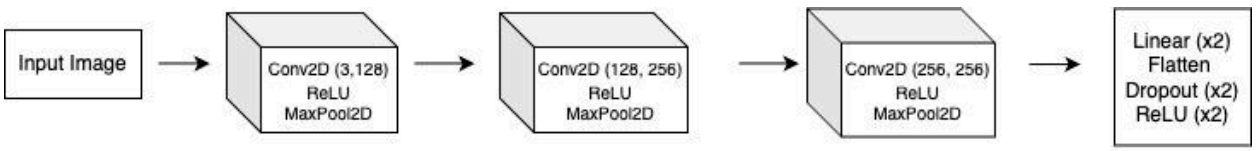
A CNN is a neural network that is most suited for image classification through its algorithm. A CNN has two blocks: the convolutional layer and the fully connected layer. The convolutional block is the part of the model that will get the certain features of the data that are used to classify the data or predict something. The image is converted into a matrix and doing some operations on them with the use of a kernel, a filter that is run over the image to make it into a new matrix of numbers. An extra row and column of padding is added on the outer layers of the matrix to give less important values a greater weight. After this, the image matrix goes through the pooling layer which removes any unnecessary information about a certain feature that was identified through the previous layers. With this, the convolutional block is completed and the data moves into the fully connected block which is, in essence, just a neural network with a couple of layers. After going through this layer, the image is identified as one of the classes stated [9].



Structure of the CNN used for the purposes of this project

AlexNet

An AlexNet is a form of a CNN that typically involves more layers and some specific layers. To be more specific, it requires a certain order of layers that work together to produce a great image classification algorithm. It was created by Alex Krizhevsky et al. in 2012 for the purpose of image classification. When first created, it was used to categorize more than one million images in the ImageNet LSVRC-2010 contest. It is composed of five convolutional layers and three fully connected layers. In the original architecture of the model, each layer had 256 kernels, but in order to minimize run time, each layer will have two to three kernels. In this contest, the AlexNet performed the best of all the entered models, showing why it is a viable contender against the basic CNN for the purposes of this experiment [10, 11, 12].



Structure of the AlexNet used in this project

Related Experiments

Both of the key concepts involved in this experiment (image classification models and federated learning) have been used before, but have not really been as interconnected as I am using them in this experiment. In many experiments, in fact, federated learning has been used with image classification models in a biological fashion as I am in this experiment.

For example, S. Wang et al. in 2024 created a model (scFed) that used federated learning in order to classify cell types. Instead of using the Flower Framework like my project does, this project creates the federated learning algorithm from scratch. This experiment also uses multiple datasets to represent multiple clients, while my experiment splits one large dataset into multiple clients. While this paper does not go over the federated learning model in comparison to centralized model, the models implemented with federated learning do seem to produce good results with all above 85% accuracy [14].

Another experiment by B. Feng et al. created a federated learning model for identifying patients with gastric cancer recurrence. This experiment also created their own federated learning algorithm, but did significantly improve results. It reduced misdiagnosis of gastric cancer recurrence by 42.23%, a significant amount [15].

Finally, one experiment by D. Chowdhury et al. actually used the Flower framework to implement federated learning. In their experiment, they used federated learning with Flower to detect Covid-19. The global model had an accuracy of 99.59% after three rounds of federated learning, showing that this form of machine learning can be very accurate [16].

These two experiments show that federated learning is not a new concept by any means, but its usage in this experiment may contribute something to the world. Not only that, but it shows that my experiment fits right into the world of computer science and artificial intelligence. However, one thing that these experiments do not include is the comparison of the federated learning model to the AI architecture when used as a centralized model. My experiment includes this vital information in the results.

Methods

To begin this experiment, a dataset containing enough data for a successful model to be created was needed. In order to find such datasets, the website Kaggle was used which is a free online community where one can find datasets and other computer science related things. After looking through many datasets for melanoma detection, I decided on the dataset called “Melanoma TFRecords 256 x 256” by Chris Deotte. This dataset was nicely split up into a train and a test folder with the train folder having 11,879 examples and the test dataset having 2000 test examples. Since this dataset was already split into a train and test folder, I did not have to split the data by myself which may have taken time to do. Additionally, each of these train and test datasets is split into folders labeled benign and malignant that represent the data [19].

In order to code the image classifiers, I used already existing models for each of them. Since the main focus of this project is not on the creation of the image classification models, I felt it was acceptable to use already existing models. For my CNN, I used the example PyTorch model given in the PyTorch documentation with a couple of tweaks. The CNN specified in the tutorial has two convolutional layers and three fully connected layers. My CNN has three convolutional layers and three fully connected layers as well as having two additional dropout

layers. The basic structure of this model was two convolutional layers, one pooling layer, and two ReLU layers [20].

With regards to the AlexNet, there is a basic structure to it that makes it so powerful and efficient. I was able to use a predefined AlexNet from kaggle.com by Paras Varshney. The AlexNet structure is composed of five convolutional layers, three pooling layers, two normalized layers, and two fully connected layers [18].

To achieve the federated learning aspect of this project, I used the established Flower framework. This framework was designed in order to make federated learning easier. The alternatives to using this framework to achieve federated learning include using TensorFlow and PySyft. The Flower framework has documentation that shows how to implement federated learning with an image classification model. To this end, for the project, all that was required was to incorporate the Flower framework with my image classification models [17].

Implementing the framework was fairly simple for my CNN, but more difficult for the AlexNet since it was defined as a Keras class, unusual to what the framework was used to. In order to solve this problem, I needed to change the implementation such that it would work with a Keras class. While most federated learning projects would require multiple clients to be actually open, the Flower framework is clever in that this is not required. It will automatically simulate this situation and from that be able to evaluate the accuracy for both the CNN and the AlexNet. This makes simulating the federated learning aspect of the project much easier, but may require it taking longer since Flower has to manually simulate a certain number of clients.

For the purpose of this experiment, the conditions for both the CNN and AlexNet simulated with federated learning were equivalent to keep the project as realistic as possible. Both models were simulated with five epochs, a batch size of 32 units, and ten simulated clients. With federated learning and the Flower framework, one can also specify the number of rounds that they want to simulate. These rounds are how many times that it will tweak some of the running conditions in order to improve the results of the model. For both models, three rounds were conducted. These conditions were used to create the best results for the experiment while not requiring too much time to run. All the training was run on Google Collab since the laptop that I have does not have its own GPU.

Results

Prior to being implemented with federated learning, the models were first individually tested for accuracy. For this testing, a batch size of 32 units and five epochs were used for both models. The reason for this testing was to determine whether or not the AlexNet was truly more accurate than the basic CNN with this dataset. The AlexNet was created in order to be more accurate for image classification so this was the determinant for the project. The results confirmed this hypothesis as the AlexNet had a testing accuracy of 93% and the basic CNN had a testing accuracy of 90%. This data set the stage for determining why one of the models may be more accurate when implemented with federated learning.

After this, the models were then implemented with federated learning using the Flower framework. This implementation took the majority of the time that was required in order to perform this experiment. In order to do this, the Flower documentation page was used. After implementation, the runtime for federated learning for both models took a significant time. Both took around two hours to run using Google Collab with each round of testing varying in time.

The AlexNet implemented with federated learning had a testing accuracy of 89.19% while the basic CNN implemented with federated learning had an accuracy of 88.89%.

While the two models could be compared in other ways, only the testing accuracy was used since that is what this project mainly tests for: whether the models differ with and without federated learning implemented.

Experimental Setup
Batch Size: 32 units Number of Epochs: 5 Federated Learning Rounds: 3 Training Data Size: 11,879 examples Testing Data Size: 2,000 examples

	Accuracy (without FL)	Accuracy (with FL)	Training Time (without FL)	Training Time (with FL)
AlexNet	93%	89.19%	~ 6 minutes	~ 2 hours
CNN	90%	88.89%	~ 13 minutes	~ 2 hours

Discussion

The results of the experiment indicate that the AlexNet was overall a better model for the purposes of this experiment. The model had higher accuracies when tested as a centralized model and when tested implemented with federated learning. The results also indicate that federated learning has results that are comparable to the results when using a centralized model. For both models, the accuracy when used with federated learning was lower by less than 5%. This indicates that federated learning can be used in place of a centralized model without sacrificing much.

The reason that the AlexNet may have had greater accuracies is because of its structure. Since the AlexNet has considerably more layers, that may be the reason that it had greater accuracies. While this may not always be the case when creating a model, it can happen that more layers correlates to greater accuracies. Sometimes more layers can result in lower accuracies because of overfitting, but in this experiment more layers produced better results.

In testing, using a lower number of clients was also tested, but this did not seem to alter the accuracy or the runtime significantly. However, changing the number of federated learning rounds does significantly reduce the runtime, but does not improve the accuracy. This indicates that the Flower framework is not impacted by changes in number of clients and can still produce comparable accuracies.

When discussing this experiment, it is important to consider potential pitfalls with the implementation. For example, the Flower framework may have not been implemented in the

best way, causing the resulting accuracies to be lower than achievable. Not only that, but maybe Flower was not the best way to implement federated learning for the purposes of this experiment. It may have been more efficient to use PySyft or even TensorFlow.

While the implementation with federated learning had similar accuracies, indicating that federated learning can provide comparable results, the runtime may lower the value of federated learning. As stated previously, the runtime with federated learning took around two hours. While this may seem like a normal amount for an AI model, the centralized models themselves took around ten to fifteen minutes to run. This is a significant gap in the runtimes for the two versions of the model. To this end, it may signify that federated learning or at least the approach used in this experiment takes significantly longer to run. This is not considering the issue of how federated learning is run when there are real clients involved. Each of the clients has to individually train a model and then send that model to the central server where the models will be aggregated. In theory, this should be slower than a centralized model since there also may be real world delays in when a client trains their model.

Using the client to client approach for real world experimentation may be even slower because it may require certain software in order for each client to receive the somewhat trained models to then train it further. Furthermore, it may be necessary to create a software that can allow for federated learning to be easily usable for hospitals in both the peer to peer and centralized approach.

Conclusion and Future Experimentation

This experiment tested the futility of federated learning when implemented with two separate models, an AlexNet and a classic CNN. These two models are considerably different with different structures, leading to reason to believe that the results of the experiment would be different. First the two structures were tested as centralized models to get a baseline for how these models would perform under normal conditions for a model. The AlexNet, as expected, performed better than the classic CNN. After this, the models were then implemented with federated learning through the Flower framework. Each of these models had around a two hour runtime, but did seem to perform comparable to the models when centralized. The results show that federated learning can indeed be used for training AI models on image classification.

Even though the results show acceptable results for using a federated learning model in comparison to a centralized model, the federated learning results can improve. For example, the specific implementation of the Flower framework that I used could be improved. Future experimentation can improve upon my Flower implementation to show that federated learning may be even better than using a centralized model not only in terms of accuracy, but also for run time. Ways that my Flower implementation can be improved is by creating a custom strategy for the specific models by changing the parameters for the federated learning model and altering the number of rounds in clients in a way to reduce run time and maximize accuracy. That is one of the major flaws that I found with federated learning when using the Flower framework: it is extremely slow. Hopefully, federated learning can be implemented in such a way to reduce the runtime by such an amount that the data privacy aspect can trump the runtime issues.

Experimental Code:

Github repository to access the code involved in this experiment for future replication. Does not include experimental data.

https://github.com/sbalaji09/FL_AlexNetCNN

References:

1. Davis, L. E., Shalin, S. C., & Tackett, A. J. (2019). Current state of melanoma diagnosis and treatment. *Cancer biology & therapy*, *20*(11), 1366–1379. https://doi.org/10.1080/15384047.2019.1640032
2. Cui, Y. Z., & Man, X. Y. (2023). Biology of melanocytes in mammals. *Frontiers in cell and developmental biology*, *11*, 1309557. https://doi.org/10.3389/fcell.2023.1309557
3. Ralli, M., Botticelli, A., Visconti, I. C., Angeletti, D., Fiore, M., Marchetti, P., Lambiase, A., de Vincentiis, M., & Greco, A. (2020). Immunotherapy in the Treatment of Metastatic Melanoma: Current Knowledge and Future Directions. *Journal of immunology research*, *2020*, 9235638. https://doi.org/10.1155/2020/9235638
4. Jordan, S., Fontaine, C., & Hendricks-Sturrup, R. (2022). Selecting Privacy-Enhancing Technologies for Managing Health Data Use. *Frontiers in public health*, *10*, 814163. https://doi.org/10.3389/fpubh.2022.814163
5. Lee, G. H., & Shin, S. Y. (2020). Federated Learning on Clinical Benchmark Data: Performance Assessment. *Journal of medical Internet research*, *22*(10), e20891. https://doi.org/10.2196/20891
6. Seyghaly, R., Garcia, J., & Masip-Bruin, X. (2024). A Comprehensive Architecture for Federated Learning-Based Smart Advertising. *Sensors (Basel, Switzerland)*, *24*(12), 3765. https://doi.org/10.3390/s24123765
7. Nadal M.C. et al.; 2021; Federated Learning: Does everyone play fair?; Madrid, Spain; https://www.mat.ucm.es/congresos/mweek/XV_Modelling_Week/Informes/Problema_2.pdf
8. Hardesty L.; April 14, 2017; Explained: Neural networks; Cambridge, Massachusetts; https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414
9. Chatterjee H.S.; July 16, 2019; A Basic Introduction to Convolution Neural Network; Saint Louis, Missouri; https://dasher.wustl.edu/chem430/readings/basic-intro-cnn.pdf
10. Kainz B.; September 22, 2020; Deep Learning - AlexNet; London, England; https://www.doc.ic.ac.uk/~bkainz/teaching/DL/notes/AlexNet.pdf
11. Bangar S.; June 24, 2022; AlexNet Architecture Explained; Trinity College Dublin; https://medium.com/@siddheshb008/alexnet-architecture-explained-b6240c528bd5
12. Krizhevsky A. et al.; 2010; ImageNet Classification with Deep Convolutional Neural Networks; Toronto, Canada; https://papers.nips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
13. Lee, J. S., & Wu, W. K. (2022). Breast Tumor Tissue Image Classification Using DIU-Net. *Sensors (Basel, Switzerland)*, *22*(24), 9838. https://doi.org/10.3390/s22249838
14. Wang, S., Shen, B., Guo, L., Shang, M., Liu, J., Sun, Q., & Shen, B. (2023). scFed: federated learning for cell type classification with scRNA-seq. *Briefings in bioinformatics*, *25*(1), bbad507. https://doi.org/10.1093/bib/bbad507
15. Feng, B., Shi, J., Huang, L., Yang, Z., Feng, S. T., Li, J., Chen, Q., Xue, H., Chen, X., Wan, C., Hu, Q., Cui, E., Chen, Y., & Long, W. (2024). Robustly federated learning model for identifying high-risk patients with postoperative gastric cancer recurrence. *Nature communications*, *15*(1), 742. https://doi.org/10.1038/s41467-024-44946-4
16. Chowdhury, D., Banerjee, S., Sannigrahi, M., Chakraborty, A., Das, A., Dey, A., & Dwivedi, A. D. (2022). Federated learning based Covid-19 detection. *Expert systems*, e13173. Advance online publication. https://doi.org/10.1111/exsy.13173
17. Flower Documentation: https://flower.ai/docs/framework/index.html
18. Raj K., (2020, September 21), [AlexNet CNN Architecture on Tensorflow (beginner)], Version 1, Retrieved July 24, 2024 from https://www.kaggle.com/code/vortexkol/alexnet-cnn-architecture-on-tensorflow-beginner
19. Deotte, C., (2018), [Melanoma TFRecords 256x256], Version 1, Retrieved July 22, 2024 from https://www.kaggle.com/datasets/cdeotte/melanoma-256x256/data

20. Chintala S., Training a Classifier, https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html#define-a-convolutional-neural-network, Accessed 22 July 2024.