**Literature Review**

**On- Device Machine Learning**

S. Dhar, J. Guo, J. (Jason) Liu, S. Tripathi, U. Kurup, and M. Shah, ‘A Survey of On-Device Machine Learning: An Algorithms and Learning Theory Perspective’, *ACM Trans. Internet Things*, vol. 2, no. 3, p. 15:1-15:49, Jul. 2021, doi: [10.1145/3450494](https://doi.org/10.1145/3450494).

Inference – a model which is trained in the cloud and is then executed on the device

This article aims to provide a survey of the current situation of Machine Learning on edge devices. It highlights how the current scenario is to resort to a cloud-based system where the edge device will collect data and conduct inference, send the data to the cloud where the model is trained and re-trained with new data from the devices. It mainly discusses the notion of having the edge devices conduct the training of the model and no cloud implementation at all for handling the model. Whilst this is an alternative approach for implementing the system for the FAIMS application which does solve the issue of being able to use machine learning to classify images whilst being offline it does also have some issues which need to be considered. The biggest constraint with having purely the client handle all the operations of Machine Learning is the resources of the device.

Cloud based systems also have issues which need to be considered security and privacy surrounding users data. Data loss, compromised cloud network.

Deep Neural Networks like CNN are the ideal algorithm for image classification as they have high accuracy for classifying the image. However due to the resource constraints of edge devices deploying models of these nature it would not be ideal as training of the model needs MBs or GBs of memory for the weights, activations, gradients, data batches making it very difficult to be deployed and trained on a mobile device. Since accuracy of the classification is a focus for the FAIMS system use of a CNN algorithm is desirable and thus have a purely on-device image classification system would not be the best approach.

V. Ganesan, ‘Machine Learning in Mobile Applications’, *International Journal of Computer Science and Mobile Computing*, vol. 11, pp. 110–118, Feb. 2022, doi: [10.47760/ijcsmc.2022.v11i02.013](https://doi.org/10.47760/ijcsmc.2022.v11i02.013).

The implementation of machine learning in mobile applications is made simpler due to the variety of frameworks that exist in the current environment. TensorFlow, Keras, Scikit, Pytorch, ML Kit, Core ML, IBM Watson, Microsoft Azure Cognitive Services, Amazon Web services, Google Cloud Machine Learning. They aim to simplify the process of creating and implementing machine learning models as they remove the need to understand the underlying approaches associated. Whilst each of these frameworks vary in their capabilities it provides a lot of options for implementing a system into the FAIMS application. Though due to the unique nature of the FAIMS application being a web application that is generated into native applications for both IOS and Android selecting the tools which cater to the already existing design effectively is desirable.

The process of machine learning for the edge device will be initially feature extraction where information from the data is taken to allow for the model to classify the image correctly. Classification is the next step where a formula is generated based on the information from the feature extraction that aims to be able to take new data inputs and classify them. These initial steps of the process are essentially the training process. Finally, prediction or scoring is evaluating how the trained model performs on new data, specifically using the metric accuracy to see how the model predicts/classifies the expected results.

When it comes to mobile applications and machine learning it seems that there are predominantly 2 common architectures that are utilised, a server-side architecture and a client-side architecture. The server-side architecture utilises a server to handle all the key operations of image classification and the mobile device will interact with the server by sending requests and the evaluated result whilst receiving from the server responses to the request. The application and the server interface by using web services to conduct the key operations meaning that the application will continuously require a connection. The client-side architecture meanwhile aims to alleviate some of the process by having model loaded onto the mobile application to conduct inference. This design of the architecture aligns closer to the use case of the FAIMS application being able to conduct ML whilst the application is offline from the server.

**FAIMS**

[1]

B. Ballsun-Stanton, S. A. Ross, A. Sobotkova, and P. Crook, ‘FAIMS Mobile: Flexible, open-source software for field research’, *SoftwareX*, vol. 7, pp. 47–52, Jan. 2018, doi: [10.1016/j.softx.2017.12.006](https://doi.org/10.1016/j.softx.2017.12.006).