Stable machine learning model for multi-carrier cellular access in Google Fi

Kehkashan Sadiq Fazal
305220750

Nikita Sivakumar
405228345

Joanna Itzel Navarro
303861189

Abstract

In this paper, we present an analysis on Google Fi in terms of the machine learning model's stability. In order to provide an in depth study on this, we advocate a novel method whereby we employ an RNN instead of the currently running linear model used in the machine learning model in Google Fi; as this seems more promising to adjust well with moving object locations. Our device-centric machine learning algorithm can also be extended to satisfy most U.S based cellular networks switching with a few alterations. We describe our proposed approach, primarily based on and tested on the machine learning model employed in Google Fi to illustrate how our scheme works. We further present preliminary results and discuss open issues.

1 Background

1.1 Introduction to Project Fi

With Google recently stepping into the world of cellular networks, a product that has caught our attention is the Google Fi. Google seeks to drive more people to use their mobile devices, where the company's services are very prominent, by providing telephone calls, SMS, and mobile broadband using cellular networks and Wi-Fi.

Google Fi uses networks operated by Sprint, T-Mobile, U.S. Cellular, and Three. Google Fi automatically switches between networks depending on signal strength and speed. It automatically connects to Wi-Fi hotspots with encryption through an automatic VPN. Phone calls transition between Wi-Fi and cellular networks. With all networks combined, the service covers more than 170 countries around the world.

1.2 Google's Algorithm for Project Fi

Phone calls will seamlessly transition to a cellular network if Wi-Fi coverage is lost. In order to achieve this smooth switching, Google uses a machine learning model, Our work in this paper aims to scrutinize this model to denote domains in the cellular spaces where Google's algorithm is costly with respect to data disruption and switching time.

1.3 Why Google's Algorithm doesn't work

We suspect that the model used by Google does not work because of a conflict of interest between the inter carrier and intra carrier selectors. As talked about in [2], picking the carrier(Sprint, T-Mobile, U.S. Cellular, and Three) is separated from picking the cell within the carrier (3G,4G,etc). This, however, suffers from the issue of persistent loops. This is illustrated with an example as follows: suppose MCSP wants a selection such that $C1, 4G > C2, 4G > C1, 3G > C2, 3G$, where C1 and C2 are two carriers. If C2 carrier has an internal intra-preference of 3G over 4G for any reason (such as better traffic), this leads to a conflict.

When looking for the carrier and cell to settle on, MCSP goes into C1 and realizes the clash in inter and intra policies, and hence, tries the next carrier and so on. This leads to a **persistent loop**.

In our project, we seek to get rid of such a loop through efficient analysis of the model and trying out techniques other than the ones used in the paper.In other words, if there is switching from $C1 \rightarrow C2 \rightarrow C3 \rightarrow \cdots Cn \rightarrow C1$, where Ci is a carrier, this is a persistent loop under the conditions of static user and policies. If it does not exist then there is stability. We seek to achieve this.

1.4 Motivation for our model: Where is Fi headed?

Fi automatically encrypts your data anytime it's connected to a network in that manner which means that no one else on the network could snoop on your connection and see what you're doing. This makes Fi a product that is promising, hence motivating us to look into the behind the scenes of the product.

According to research, Fi tends to be best for people who use a relatively small amount of mobile data. This could probably be the case because of the instability in switching from one carrier to another which occurs when Fi attempts to switch to the "better" carrier. We seek to explore the machine learning techniques employed by Google in Google Fi to see if we can stabilize this transition from one carrier to another in a way that does not lead to loss of connectivity and hence, motivates a larger portion of the internet users to use Fi.

2 Proposed Solution

The goal of this project is to analyze the machine learning model used in Google Fi and propose a solution to contrast their current model for carrier switching, which might perform better under set conditions. For this purpose, we propose the following:

2.1 Analyze and Reverse Engineer Google's current code

Analyze the machine learning code used in Project Fi by setting the current implementation up, and reverse engineering it to obtain the playground for us to get our hands dirty. Run the code and attempt switching between carriers to obtain results and demarcate expected outcomes to surpass.

2.2 Do targeted emulation

Replace the existing machine learning model implemented linearly, with another model which implements RNN and perform controlled tests on it. We will be using Tensorflow, Java and Python for this purpose.

2.3 Propose new design

We propose using a Recurrent Neural Network (RNN) in place of the current linear model that Google Fi uses as a way to improve multi-carrier switching.

From paper [2] we gather that there is an assumption of static users who do not move, and performance metrics that do not change (deterministic policies). This causes several problems, one of which is described in Section 3.3. We plan to take dynamic settings into account to mitigate some of the problems like that of persistent loops. If we do assume dynamic users, a new problem of transient loops will arise. These will also need to be dealt with.

RNNs can deal with these issues in an elegant manner. While RNNs are traditionally difficult to train, with a Long Short-Term Memory, or LSTM, network, which is perhaps the most successful RNN, it overcomes the problems of training a recurrent network and in turn has been used on a wide range of applications. We plan on using an LSTM or some other form of RNN for our stability check on the carrier switching model implemented in Project-Fi. RNNs in general and LSTMs in particular have received the most success when working with sequences of information, that is fast paced and regularly changing.

RNNs are used for text data, speech data, classification prediction problems, regression prediction problems and generative models. We believe that our problem can be moulded to fit under the classification prediction problem and hence, this might be a good solution for the problem.

The main drawbacks of recurrent neural networks is that they are not appropriate for tabular datasets as you would see in a CSV file or spreadsheet. They are also not appropriate for image data input. Here, in our project, we face no such issues.

2.4 Get access to cellular coverage data

After successful emulation with generated inputs, we plan to either use real cellular data or crawl coverage data to perform analysis on and further corroborate our claims for using RNN. We will also be evaluating the cost of carrier switching and the time cost of training the model.

2.5 Evaluation

Finally, we plan to run tests to evaluate the merit of our model versus Google's model using cellular data. This evaluation will help us determine the feasibility of our solution and make some changes if required.

3 Timeline

Week 2	Review project description and conduct literature review
Week 3	Submit proposal and create group GitHub
Week 4	Examine Google's algorithm; do reverse engineering, read code used for model currently implemented
Week 5	Begin using TensorFlow to run code
Weeks 6-7	Refine machine learning algorithm(s) and update code
Week 8	Conduct targeted emulation to test our proposed design
Week 9	Collect cellular data to measure model performance against Google's
Week 10	Finish write-up of results and submit final paper

4 Literature Survey

4.1 Motivation for the merger of machine learning and cellular spaces

In [1], closed network operations are studied. Issues on mobile network analysis and problems with conventional methods are explored. Results from this paper suggest that a two-level, device-centric machine learning approach yields better results; specifically, a more open system and fine-grained analysis. Typically, there are 4 main problems that arise which make network analysis challenging over fine-grained networks:

1. tightly-guarded system operations;
2. access barriers to hardware and software stack;
3. complexity in data and control planes (as can be seen from Fig. 1 and 2);
4. distributed operations across multiple protocol layers

To address these challenges and other challenges presented with mobile networks, a device-centric machine learning approach was taken by Zengwen Yuan, et al, which results in various latency components being revealed compared to other methods. With refining the two-level machine learning approach used by the authors in this paper, issues behind the 4G/5G mobile network issues can be addressed.

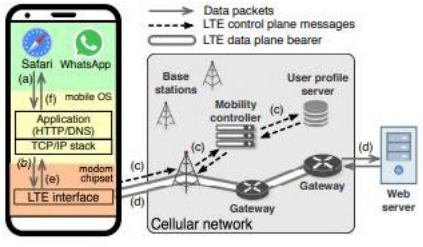

Figure 1: Mobile apps access servers via LTE network [1]

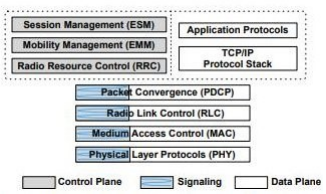

Figure 2: LTE protocol stack for both control and data planes. [1]

4.2 Machine learning tools available for Cellular Networks

According to the work in [3], TensorFlow3 is a machine learning library developed by Google. It enables deploying computation graphs on CPUs, GPUs, and even mobile devices, allowing ML implementation on both single and distributed architectures. Although originally designed for ML and deep neural networks applications, TensorFlow is also suitable for other data-driven research purposes as well. Detailed documentation and tutorials for Python exist, while other programming languages such as C, Java, and Go are also supported. Currently it is the most popular deep learning library. Building upon TensorFlow, several dedicated deep learning toolboxes were released to provide higher-level programming interfaces, including Keras4 , Luminoth 5 and TensorLayer. The paper also provided a high level summary of state-of-the-art deep learning architectures, particularly highlighting ones built upon other models as seen in Fig. 3.

Model	Learning scenarios	Example architectures	Suitable problems	Pros	Cons	Potential applications in mobile networks
MLP	Supervised, unsupervised, reinforcement	ANN, AdaNet [139]	Modeling data with simple correlations	Naive structure and straightforward to build	High complexity, modest performance and slow convergence	Modeling multi-attribute mobile data; auxiliary or component of other deep architectures
RBM	Unsupervised	DBN [140], Convolutional DBN [141]	Extracting robust representations	Can generate virtual samples	Difficult to train well	Learning representations from unlabeled mobile data; model weight initialization; network flow prediction
AE	Unsupervised	DAE [142], VAE [143]	Learning sparse and compact representations	Powerful and effective unsupervised learning	Expensive to pretrain with big data	model weight initialization; mobile data dimension reduction; mobile anomaly detection
CNN	Supervised, unsupervised, reinforcement	AlexNet [85], ResNet [144], 3D-ConvNet [145], GoogLeNet [129], DenseNet [146]	Spatial data modeling	Weight sharing; affine invariance	High computational cost; challenging to find optimal hyper-parameters; requires deep structures for complex tasks	Spatial mobile data analysis
RNN	Supervised, unsupervised, reinforcement	LSTM [147], Attention based RNN [148], ConvLSTM [149]	Sequential data modeling	Expertise in capturing temporal dependencies	High model complexity; gradient vanishing and exploding problems	Individual traffic flow analysis; network-wide (spatio-) temporal data modeling
GAN	Unsupervised	WGAN [78], LS-GAN [150]	Data generation	Can produce lifelike artifacts from a target distribution	Training process is unstable (convergence difficult)	Virtual mobile data generation; assisting supervised learning tasks in network data analysis
DRL	Reinforcement	DQN [19], Deep Policy Gradient [151], A3C [77], Rainbow [152], DPPO [153]	Control problems with high-dimensional inputs	Ideal for high-dimensional environment modeling	Slow in terms of convergence	Mobile network control and management.

Figure 3: Summary of different deep learning architectures [3]

The rest of the paper is structured as follows. Section II introduces Related works. Section III discusses machine learning based approach. It further sketches out that the proposed scheme works better in terms of stability while switching carriers. Section IV delineates the detailed procedure on applying the approach on the case study of stability analysis. Section V presents the preliminary analysis results by using the procedure and section VI discusses the possible solutions and future issues. Finally, Section VII concludes the paper.

4.3 Policy Conflicts in Multi-Carrier Cellular Access

In paper [2] policy based switching is introduced. Here, picking the carrier is separated from picking the cell, so it does not need fine grain details of how the cell is selected within the carrier. This, however, suffers from the issue of persistent loops as discussed in the Background section 1.3.

References

[1] Yuan, Zengwen Li, Yuanjie Peng, Chunyi Lu, Songwu Deng, Haotian Tan, Zhaowei Raza, and Taqi. Machine learning based approach to mobile network analysis. *International Conference on Computer Communication and Networks (ICCCN)*, 2018.

[2] Zengwen Yuan, Qianru Li, Yuanjie Li, Songwu Lu, Chunyi Peng, and George Varghese. Resolving Policy Conflicts in Multi-Carrier Cellular Access. *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18). ACM, New York, NY, USA*, 2018.

[3] Zhan, Patras, and Haddadi. Deep learning in mobile and wireless networking: A survey corr. *arXiv preprint*, 2018.