Flow–based Neural Network Development with the Help of Edge Computing

Name: Ju Pan \
Email: pjokk722@email.arizona.edu

1. Abstract

This project focuses on the combination of deep neural network and edge computing. The project will briefly introduce the deep neural network and the edge computing. Then some existing works will be discussed and the tradeoffs will be evaluated. After that, some new ideas about improving the deep neural network with the help of edge computing in some specific use cases will be presented. The project also talks about the possibility of leveraging the new Internet architecture Named Data Networking (NDN) in edge computing. Unfortunately, there is not enough time and resources for me to implement the proposed system this semester but the details of the implementation plan will be discussed. Besides, the evaluation plan will be discussed as well. I've been focusing on reviewing papers, brainstorming the ideas and implement the NDN feature to support the async applications (https://redmine.named-data.net/issues/4931).

2. Introduction

Deep neural network is widely applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance. Most of these deep neural networks are deployed on cloud servers which is quite far from the end users so that we may face challenges like high latency, lack of bandwidth, and user data privacy issues. To relieve these challenges, edge computing may be worth some research.

Edge computing is a distributed computing paradigm which brings computation and data storage closer to the location where it is needed, to improve response times, save bandwidth and enhance data privacy. If we can split deep neural network into multiple groups/modules and offload some of the groups/modules from the cloud to the edge nodes which are closer to the end users, we may end up with acquiring some benefits like reduced latency and bandwidth. Besides, with the raw data being processed at the edge, the centralized cloud won't have a comprehensive knowledge of the user data so the user data privacy may be protected as well.

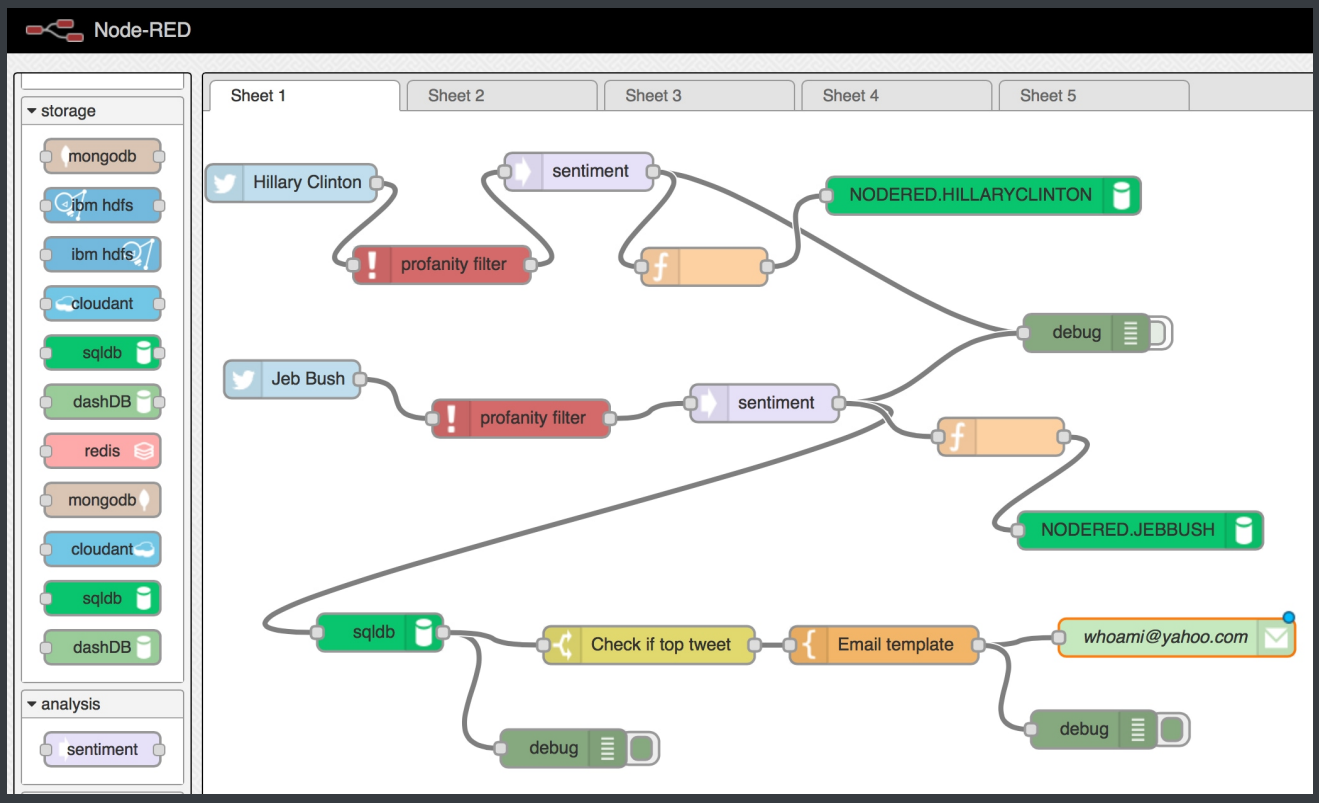
To combine deep neural networks with edge computing, there are some questions to be answered properly:

1. Is deep neural network separable? If we can't find a way to split the deep neural network, the entire project may not be viable.
2. If a deep neural network is separable, is there any global rules to follow separate it and make the most benefit of applying the edge computing? If we can't find an optimal way to improve any aspects of the deep neural network, the project doesn't make any sense either.
3. How much improvement can we generate from utilizing edge computing here? And what are the use cases that may benefit from this edge-computing-enhanced deep neural network? We do care about the applications, we don't want to invent something that nobody could benefit from.

Those questions lead to some corresponding discussions in the following sections. The key is that we want to have a useable and useful edge-computing-enhanced deep neural network.

We will also apply flow-based programming concept to this project. [2] In computer programming, flow-based programming (FBP) is a programming paradigm that defines applications as networks of "black box" processes, which exchange data across predefined connections by message passing, where the connections are specified externally to the processes. These black box processes can be reconnected endlessly to form different applications without having to be changed internally. With the help of flow-based programming, someone who doesn't have much machine learning knowledge and computer networking knowledge can easily develop an deep neural network application by wiring some components together, they only need to care about the input and the output of these components and their domain knowledge, here is a figure showing what a flow-based application looks like. The flow itself is quite self-explanatory, this is an application that analyze the sentiment of twitters about Hillary Clinton and Jeb Bush.

The project also plans to evaluate our application on TCP/IP and NDN (Named-data Networking) protocols to see what difference will they make in terms of the latency and the network bandwidth. The core innovation in Information Centric Networking (ICN) including Named Data Networking (NDN) is the use of application-level names as the identifier for network packets. This allows the network to identify data independent of a particular connection, and to retrieve desired data from anywhere instead of a single given host. NDN network also supports in-network caching and built-in security. NDN directly secures the data in layer2 which is more convenient and secure for the application developer.



3. Related work

With an significantly increasing amount of Internet of Things (IoT) devices, it's reasonable to expect that most of data generated by these IoT devices should be processed locally by the devices themselves or at the edge nodes. Otherwise, the network bandwidth may be overwhelmed badly. In [1], authors introduce a face recognition application that processes the captured photos at the edge as opposed to the cloud. [4] BranchyNet proposed a solution of classifying samples at earlier points in a neural network, called early exit points, through the use of an entropy-based confidence criteria. If at an early exit point a sample is deemed confident based on the entropy of the computed probability vector for target classes, then it is classified and no further computation is performed by the higher NN layers. [3] Introduces a distributed deep networks over the cloud, the edge and edge devices. A DDNN allows fast and localized inference using shallow portions of the neural network at the edge and end devices. By training DDNN end-to-end, the network optimally configures lower NN layers to support local inference at end devices, and higher NN layers in the cloud to improve overall classification accuracy of the system.

4. Architectures

Here are three types of deep neural network application paradigm nowadays. Figure 2 is the case that the whole deep neural network running on the user local machine. In this case, the computing efficiency may be varies among different kinds of devices. And the power consumption will be a big concern if the user device is mobile.

Figure 3 describes the cloud computing scenario. In this case, the deep neural network is put on the cloud, when the user capture the raw data using the device, the raw data will be transmitted all the way to the cloud server and the classification will be done there as well. Then the final result will be sent back to the local device. This paradigm doesn't require the local device to have much computing resource and power consumption is relatively low too. However, the potential problems with this paradigm are data privacy issue and high bandwidth usage and high latency. Since the user is sending all the raw data to the cloud, the service provider will have the full knowledge of your data content, this may cause issue for example when the service provide sell your data to some third-party organizations. Sending all the raw data through the network may also takes up much bandwidths and because of the long distance between the local device and the cloud server, the user may experience a high latency to get the final result back.

Figure 4 represents the edge computing paradigm. In this scenario, we try to offload some of the services from the cloud server to the edge devices. In our case, we try to put part of the deep neural network on the edge devices and the rest still stay on the cloud server. We can apply some early exist logic to the deep neural network, if the classifier is confident enough about the input data at the early stage of the deep neural network, and assuming this early stage stays at the edge devices, the edge device can immediately send back the final result. Otherwise, the processed data will be transmitted to the cloud server and finish the rest of the classification. In this case, the bandwidth can be reduces since we are not sending all the raw data through the network to the cloud server and the data privacy can be protected as well because we are sending the processed data instead of the raw data, it's almost impossible for the service provider to figure out the practical meaning of these processed data. Besides, since the result may be produced early by the edge devices, the user may experience a lower latency. One challenge of applying edge computing paradigm to deep neural network is how do we separate the deep neural network - which layers should be put on the cloud, which layers should be put on the edge devices and which layers should stay locally on the user devices. For this project, to evaluate the architures with different internet protocol , we plan to reuse the idea in [3].

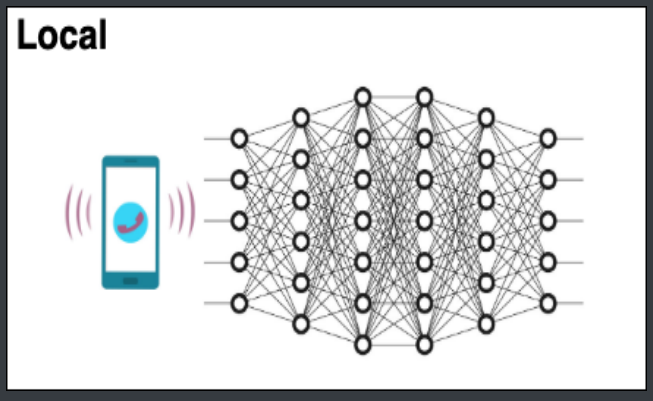


Figure 2. Local Computing

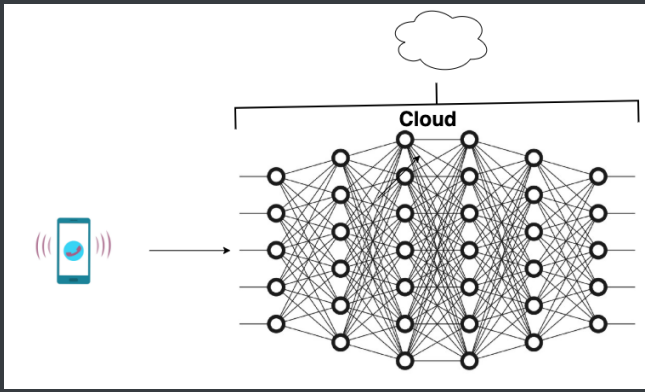


Figure 3. Cloud Computing Paradigm

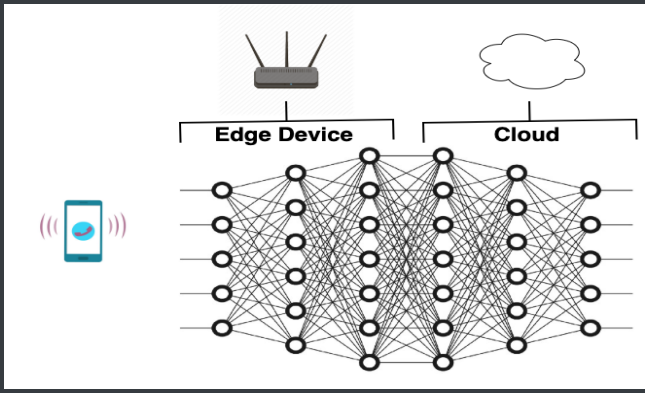


Figure 4. Edge Computing Paradigm

5. Implementation and evaluation plan

I actually didn't implement any of the following plans this semester. The main reasons are: 1. we are working on a new NDN feature in our NDN codebase to support any async application, It's important to have this feature ready first, then we can test on our ideas; 2. I don't have any funding to setup a cloud server on AWS or GCP and there is something wrong with our lab server, we don't have access to the lab to fix it due to the special situation. So, I want to apologize for this incomplete project, but I will finish them as long as everything goes back to normal.

The deep neural network application that I am going to use is face recognition. I'll follow the implementation of [3] to achieve the early exit functionality - if the classifier is confident enough, then it can exit earlier without any further processing.

I plan to use Samsung S9+ Android phone as the local device, Macbook Pro as the edge device and a ubuntu server on AWS as the cloud server. On the Android device, I'll write a simple picture capturing application in Java and also implement the face recognition functionality inside the app. But this face recognition functionality may not be used when we are testing edge computing and cloud computing. The app will provide computing options - local computing, edge computing, and cloud computing. I'll implement both the TCP/IP version and the NDN version of the applications.

The metrics that I am going to evaluate are:

1. Network bandwidth
2. Latency - how long does the user need to wait to get the result back
3. Classification accuracy
4. How many times does the edge devices directly return back the result (early exit)
5. The energy consumption of the local device (we can monitor the application power consumption in Android studio)

Here are the test cases I want to try:

1. Run the deep neural network locally
2. Put the first part of the deep neural network on edge device, and the rest on the cloud server, then use the same picture we used in test1.
3. Try different ways to splitting the deep neural network.
4. Put the entire deep neural network on the cloud.
5. Use the NDN as the internet protocol and try all the test again.

After evaluating all the test cases in terms of aforementioned metrics, I'll try to implement the whole app in NodeRED (the flow-based programming platform). NodeRED is a javascript-based programming platform, so I need to refactor all the code I have into Javascript. The goal of using NodeRED is to make people without programming knowledge easily develop the application they want. I've already played with NodeRED in this semester and made a simple face recognition program using Opencv.

6. Conclusion and Future Work

This work is more about the review and to-dos. This work mainly has three parts: 1. Separating the deep neural network and offload some layers from the cloud to the edge; 2. Implementing the network architecture using Named-data Networking; 3. Implementing everything on NodeRED to make the development easier for general public. The future work is aforementioned, I need to implement a face recognition base application and change the network protocol to NDN, then I need to do the aforementioned experiments to do the evaluation. Finally I'll port the program to NodeRED to make a flow-based programming application. I can't say if my work will perform better then vanilla cloud-computing version of the project at this moment or what should be done to make it works better, but we will know later.

7. Acknowledgement

I want to thank Dr. Carlos Scheidegger for his interesting classes. He explained everything clearly to the whole class. I have no background on ML before this class, after this semester's study, I think I understand the basics. I am working on computer networking, I was always wondering is there any chance that I can combine computer networking and machine learning together. Dr. Carlos Scheidegger's course inspired me of doing this project. Although I couldn't actually finish the whole project this semester, I wish I could continue working on this research topic in the future.

X. Reference

[1] S. Yi, Z. Hao, Z. Qin, Q. Li, "Fog computing: Platform and applications", Hot Topics in Web Systems and Technologies (HotWeb) 2015 Third IEEE Workshop, pp. 73-78, 2015. \
[2] https://en.wikipedia.org/wiki/Flow-based_programming \
[3] S. Teerapittayanon, B. McDanel, H. Kung, "Distributed Deep Neural Networks over the Cloud the Edge and End Devices", Proc. IEEE 37th Int'l Conf. ICDCS, 2017. \
[4] S. Teerapittayanon, B. McDanel, H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks", International Conference on Pattern Recognition, 2016. \
[5] T. Liang, J. Pan and B. Zhang, "NDNizing Existing Applications: Research Issues and Experiences", Proc. 5th ACM Conference on Information-Centric Networking (ICN), 2018.