CHAPTER 1

INTRODUCTION

Our search queries, browsing history, purchase transactions, the videos we watch and our movies preferences are but few types of information that are being collected and stored on daily basis. This data collection happens within our mobile devices and computers, on the streets, even in our own offices and homes. Such private data is being used for a variety of machine learning applications.

Machine Learning (ML) techniques have begun to dominate data analytics applications and services. Recommendation systems are the driving force of online service providers such as Amazon, Netflix and Spotify. Finance analytics has quickly adopted ML to harness large volume of data in such areas as fraud detection, risk-management, and compliance. Deep Neural Network (DNN) is the technology behind voice-based personal assistance, selfdriving cars, automatic image processing, etc. By deploying ML technologies to cloud computing infrastructures, they are benefiting numerous aspects of our daily life.

However, the surge of ML is accompanied by a public concern of personal data privacy. The basic business model of ML-based data analytics is a closed circle: with more data, more accurate model can be trained, which means more useful services and more users, and they finally lead to more data being collected. At the same time, people are increasingly aware of the data privacy issue. Ubiquity of sensing via mobile and IoT devices has caused a surge in personal data generation and use. They contain our photos, browsing history, and voice records, etc., some of which one might not want to share with data analytics service providers. But these data are also perfect targets for them to collect in order to provide personalized services. So even with legal regulatory frameworks such as EU's General Data Protection Regulation (European-Commission 2016), the aforementioned business model cycle is still difficult to break.

Data Security and Privacy requirements

No matter whether it is cloud computing or edge computing, the end user's privacy data needs to be partially or completely outsourced to third parties (such as cloud data center or edge data centers), and its ownership and control are separated, which will easily lead to data

loss, data leakage, illegal data operations (replication, publishing, dissemination) and other data security issues, data confidentiality and integrity cannot be guaranteed. Therefore, the security of outsourcing data is still a fundamental problem of edge computing data security.

- Confidentiality: The confidentiality is a fundamental requirement that ensures only data owner and user(s) could access the private information in the edge computing. It prevents unauthorized parties access to the data when the users' private data is transmitted and received in edge or core network infrastructure, and stored or processed in edge or cloud data center.
- Integrity: The integrity is under an obligation to ensure the correct and consistent delivery of data to the authorized user(s) without any undetected modification of the data. The absence of integrity auditing measures could affect the users' privacy.
- Availability: The availability ensures that all the authorized parties are able to access the edge and cloud services at any places as per users' requirements. In particular, it also means that the users' data which stored in edge or cloud data center with cipher text form, can be processed under different operational requirements.
- Authentication and access control: The authentication ensures the identity of a user is authorized which means it is a process of establishing proof of user's identities. Furthermore, the access control acts like a bridging point of all the security and privacy requirements by the control strategy, it determines who can access the resources (authentication) and what kind of actions can perform such as reading (confidentiality) and writing (integrity).
- Privacy requirement: The security mechanisms are used to guarantee that all the outsourcing information of users, such as data, personal identity, and location, to be secret under the honest but curious adversaries. In addition, the data security mechanisms, like encryption, integrity auditing, authentication and access control, can preserve the privacy of the users directly or indirectly in edge computing.

Existing solutions that aim at solving this data privacy issue mostly focus on improving the cloud-based services, including making databases more secure, hiding trained ML models enable users to choose which part of data to remove before uploading to cloud servers. In this case, the

centralized cloud computing model has shown the inherent problems, which can be summarized as follows:

1) Linear growth computing capabilities of cloud computing cannot meet the multi-sources data processing requirements of massive data at the edge of network;

2) The network bandwidth and the transmission speed have come to a bottleneck because of the large scale of user access, while the long distance transmission between user and cloud center will lead to the high service latency and waste of computing resources

To overcome the drawbacks of data analytics on cloud, we are proposing this data analytics on edge devices. The following are the objectives of this proposed model.

- To deploy ML services on edge devices. Moving services from cloud to users' edge devices can keep the data private, and effectively reduce the communication cost and response latency.
- To maintain scalability of analytics. As the number of sensors and devices grow, the amount of data that they collect also grows exponentially and it increases the strain on the central data analytics resources to process these huge amounts of data. Edge analytics enables organizations to scale their processing and analytics capabilities by decentralizing to the sites where the data is actually collected.
- To reduce the overall expenses by minimizing bandwidth, scaling of the operations and reducing the latency of the critical decisions.

In this paper, I further identify two other main challenges to do ML based data analytics on edge devices. The first one is the lack of suitable models for users. Training a model requires large datasets and rich computing resources, which are often not available to most users. That's one of the reasons that they are bounded to use the models or services provided by large companies.

Towards this end I propose the idea "composable services", where the users can construct new services based on existing ones and contribute them to the community. The second new challenge is the deployment of services. Users may lack required knowledge of ML to understand how a model works and how to properly deploy it to any local devices. I present the design of a novel system Zoo to support the construction, compose, and easy deployment of ML

models on edge and local devices. In the deployment example, ML services are proved to be easy to compose and deploy with Zoo. Our evaluations show its performance compared with state-of-art deep learning platforms and Google ML services.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing System

Existing solutions of privacy preserving techniques uses encryption techniques and cloud based analytics. Many online service providers, while collecting large scale data from users, may be prone to data breaches.

Identity-Based Encryption

The identity-based encryption was proposed by Shamir as a simplification scheme of certificate management in e-mail systems. This scheme enables any pair of users to communicate securely and to verify each other's signatures without exchanging private or public key, without keeping key directories, and without utilizing the services of a third party. The IBE scheme allows users to select an arbitrary string that provides a unique identity for him to the other party as a public key, compared with the traditional Public Key Infrastructure (PKI) technology, the users' private key in IBE is generated by using a Private Key Generator (PKG) instead of by a public Certificate Authority (CA) or users. The IBE scheme includes three main phases:

1) Encryption: When Alice sends an email to Bob, the email will be encrypted by Bob's email address (bob@b.com) as the public key

2) Identity authentication: Upon Bob received the encrypted email, he needs to authentication himself and gets private key from the PKG

3) Decryption: Bob decrypts the encrypted email and get the messages.

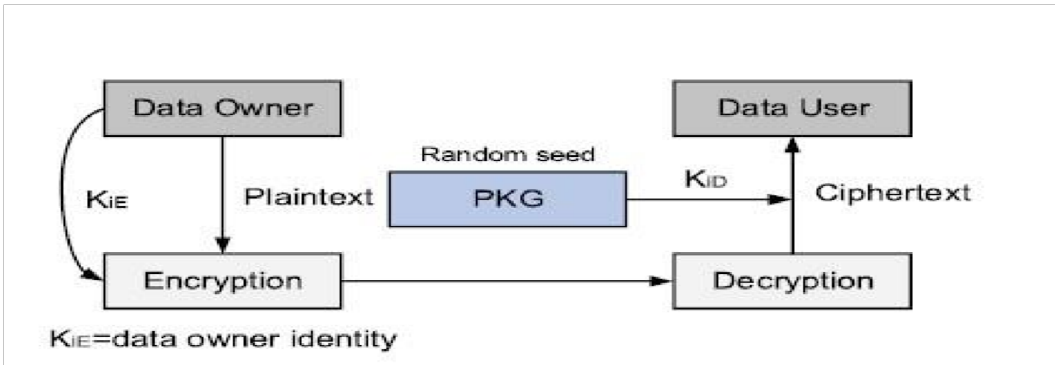

Figure 2.1: IBE scheme

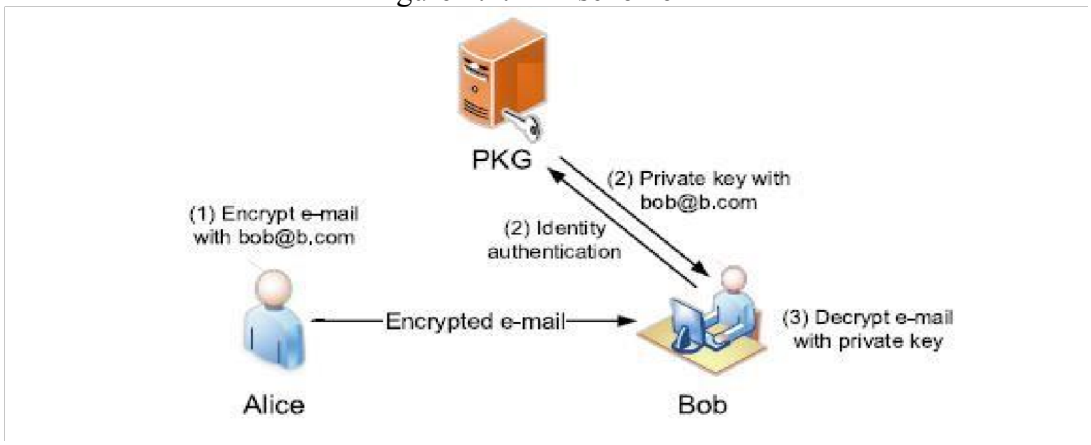

Figure 2.2: E-mail system based on IBE

The basic IBE scheme proposed by Shamir exists two problems that cannot be ignored:

1)How can Bob prove his identity to multiple trusted third parties?

2) How can a trusted third party securely send Bob's private key to Bob's hand?

Attribute-Based Encryption

Attribute-based encryption (ABE) is a cryptographic primitive to control the decryption ability of the data owner over the encrypted data. An attribute-based access control system consists of two entities: 1) Trusted authority (TA) who is in charge of publishing attribute keys and managing users attribute set, 2) The user includes the message sender and the receiver which correspond to the data owner and user.

Attribute-Based Encryption (fuzzy IBE) as a re-construction of IBE scheme in which the identities are replaced of a set attributes. In ABE scheme, each attribute of the user is mapped to the Z^*p by the hash function, which the cipher text and secret keys are related to the attributes. The basic ABE scheme can only represent the ``threshold" operation of the attribute, which the

threshold parameter is set by the authority. As a result of this feature, the access control policy is totally determined by the third party, which may cause the privilege abused, service manipulation and privacy leakage. In many practical applications, they need flexible access control policies to support AND-OR-INVERT and threshold operations for attributes, so that the data sender can specify the access control policies when encrypting data.

The basic ABE scheme was improved as two main types, as follows:

1. Key-Policy Attribute-Based Encryption (KP-ABE)was proposed by Goyal *et al*. [73] based on the monotonic access structure which consists only of AND, and OR gates.

and non-interactive cryptographic assumptions in the standard model by Walter..

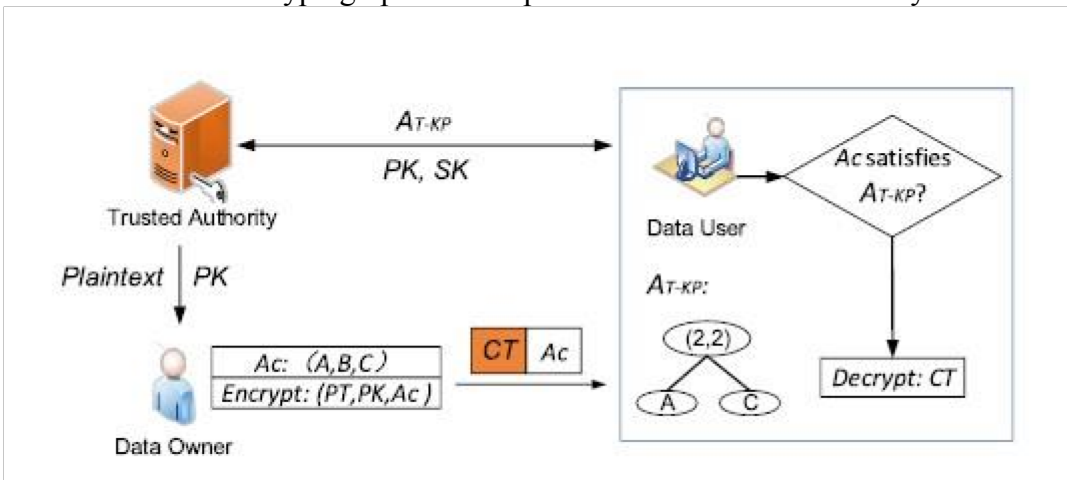

Figure 2.3: ABE scheme

Privacy-preserving Analytics on Cloud

Many popular data analytics are conducted on cloud, therefore its data privacy issue has attracted a lot of research interests. Making databases private and secure is one of the solutions. (Babuji et al. 2016) introduces a cloud-based framework that enables secure management and analysis of large, and potentially sensitive, datasets. It claims to ensure secure data storage that can only be accessed by authorized users.(Joshi et al. 2016) has developed an ontology so that big data analytics consumers can write data privacy policies using formal policy languages and build automated systems for compliance validation.

Hiding data alone is not enough. The model itself can also reveal private information. (Fredrikson, Jha, and Ristenpart2015) has shown that, with access to the face recognition ML

model, the authors can use that to recover recognizable images. (Abadi et al. 2016) develops a method that can prevent these kinds of model inversion attacks against a strong adversary who has full knowledge of the training mechanism and access to the model parameters. Similarly,

(Papernot etal. 2016) proposes a "teacher-student" approach that is based on the knowledge aggregation and transfer technique, so as to hide the models trained on sensitive data in a black box.

Instead of hiding data or models, some research suggest user to choose which part of data to upload. (Xu et al. 2017) proposes a mechanism to allow users to clean their data before uploading them to process. It allows for prediction of the desired information, while hiding confidential information that client want to keep private. RAPPOR (Erlingsson, Pihur, and Korolova 2014) enables collecting statistics data from end-user in privacy-preserving crowdsourcing.

However, all of the afore mentioned work focus on the traditional cloud side solution.

Users' data are still collected to central server for processing, which are prone to issues such as increased service response latency, communication cost, and single point failure.

2.2 Proposed System

This proposed system is used to overcome the problems associated with the existing system.

This proposed system is based on data analytics on edge devices which is commonly known as Edge analytics. Edge analytics refers to the analysis of data from some non-central point in a system such as connected devices. As an emerging term, it defines the attempt to collect data in decentralized environments. This has been a popular way to drive analytics, but now scientists are exploring how edge analytics can work as an effective alternate option. In some ways it goes along with the Internet of things. It has been using as an alternative to traditional analytics which is performed in centralized ways.

Data Analytics on Edge Devices

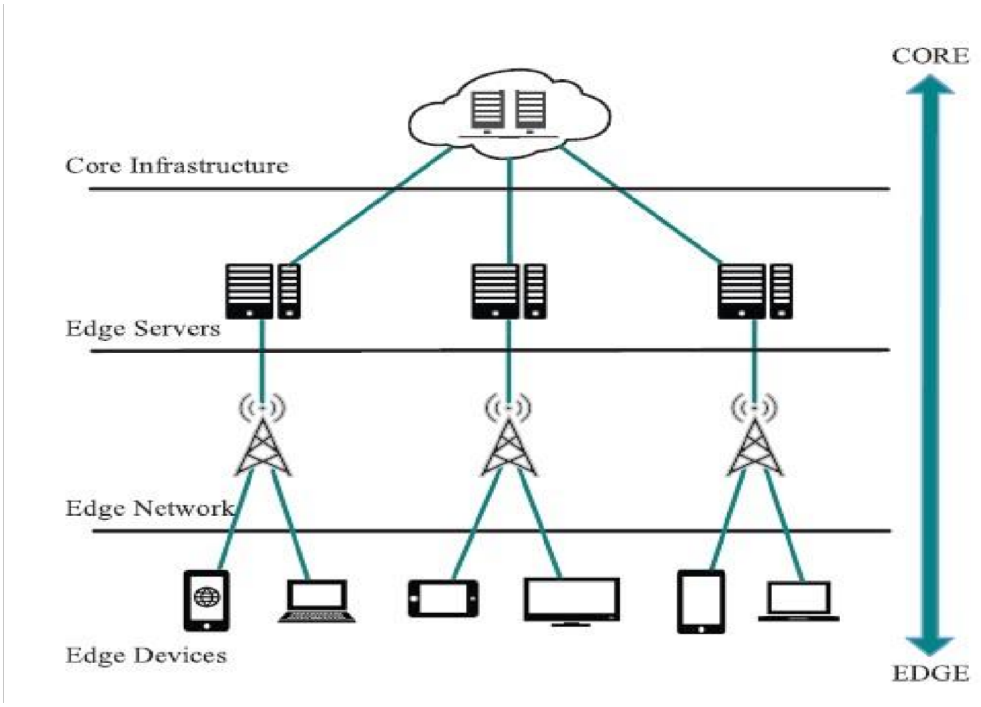

Figure 2.4: Architecture of edge analytics

The edge devices played as active participants in the distributed edge environment at different layers, so that even small portion compromised edge devices could lead to harmful results for the whole edge ecosystem. For example, any devices manipulated by an adversary can try to disrupt the services with the injection of false information or intrude the system with some malicious activities. In addition, malicious devices can manipulate services in some particular scenarios, where the malicious adversaries have gained the control privilege of one of these devices.

Computing on edge and mobile devices has gained rapid growth. Recently HUAWEI has identified speed and responsiveness of native AI processing on mobile devices as the key to a new era in smartphone innovation (Huawei-News 2017). Applications in this field are of great interest to academia and industry. (Mortier et al. 2016) provides a collection of software components that enable individual data subjects to manage, log and audit access to their data by other parties. (Pan et al. 2017) presents an environment monitoring smartphone app that allow users to takes photos of the outdoor to recognize air quality. Intrusion Detection System (IDS) is important in securing computer networks. Using ML for IDS is especially suitable in stopping

attacks that do not have to know signature. It achieves high detection accuracy and low energy consumption.

More and more advanced smart services are being pushed to edge devices. Intel's Movidius Neural Compute Stick (Movidius 2017) is a tiny deep learning device that one can use to accelerate AI programming and DNN inference application deployment at the edge. Kvasir (Wang et al.2016b) is a semantic recommendation system built on top of latent semantic analysis and other state-of-the-art data analytics technologies. Specifically, it seamlessly integrates an automated and proactive content provision service into web browser on users' end.

Many challenges arise when moving ML analytics from cloud to edge devices. One is that compared with resource rich computing clusters, edge and mobile devices only have quite limited computation power and working memory. To accommodate heavy ML computation on edge devices, one solution is to train suitable small models to do inference on mobile devices.

This method leads to unsatisfactory accuracy and user experiences.

The method of training personalized model on local devices from an initial shared model. Instead of moving data from user to cloud, our method provides for model training and inference in a system where computation is moved to the data. Specifically, I take an initial model learnt from a small set of users and retrain it locally using data from a single user. It is proved to both be robust against adversarial attacks and can improve accuracy.

CHAPTER 3

SYSTEM DESIGN AND IMPLEMENTATION

The basic idea of this system is to perform data analytics on edge devices. This proposed model infers that users should not have to construct new ML services every time new application requirements arise. In fact, many services can be composed from basic ML services.

We define that a ML service consists of two parts: development and deployment. Development is the design of interaction workflow and the computational functions of different ML services. We provide primitives to construct complex services from simple ones. For example, one of the most important primitives is sequential connection, where the output of one service is used as input of another service.

The proposed system designs model called “Zoo system”. The Zoo system aims at providing user-centric, ML-based services that enables service pulling, composing, sharing, and compatibility checking. The system architecture of Zoo is shown in Figure 3.1.

In Figure 3.1, a user first designs the target service on computer C (Step 1) in a configuration file, and executes the compose script with Zoo. The square, hexagon, etc. here represent different existing ML services. They are connected with the sequential connection primitive in the new structure design. The local repository is first checked to see if models of these required services are cached locally. If not, they will be first pulled from remote repositories. In our current implementation, all published models are stored and accessed from Github Gist (represented by Server A), but this approach could be extended to support pulling services from other peer devices such as machine B in the figure (Step 2).

Deployment deals with service interface and location definition. It is separated from the logic of service construction. Users can move services from being local to remote and vice versa, without changing the structure of constructed service. Deployment is not limited to edge devices (machine E), but can also be on cloud servers (server D), or a hybrid of both cases, to minimize the data revealed to the cloud and the associated communication costs (Step 3). Thus by this design a data analytics service can easily distributed to multiple devices. Finally, user can contribute newly created services to the model repositories so that it can be accessed by others(Step 4).

The Zoo system is implemented on Owl , an open-source numerical computing system in OCaml language. The reason we choose Owl to support the implementation of Zoo is some of its nice features. Owl provides a full stack support for numerical methods, scientific computing, advanced data analytics on OCaml. Built on the core data structure of matrix and n-dimensional array, Owl supports a comprehensive set of classic analytics such as math functions, statistics, linear algebra, as well as advanced analytics techniques, namely optimization, algorithmic differentiation, and regression. On top of them, Owl provides Neural Network and Natural Language Processing modules. Zoo relies on these modules to construct basic models. It has static type checking, and Owl's ML modules have shown great expressiveness and code flexibility. Moreover, Owl can use the parallel and distributed engine at lower level to support distributed numerical computing and data analytics. It supports different protocols and multiple synchronization techniques, which is a crucial building block for collaboration between network nodes, and thus important for scaling up data analytics in heterogeneous edge networks.

Deployment Example

Next we show an example to use Zoo to deploy an image classification services based on InceptionV3 neural network architecture. The service is deployed on a representative resource constrained personal device: a Raspberry Pi 3Model B.

InceptionV3 is one of Google's latest efforts to do image recognition. It is trained for the Image Net Large Visual Recognition Challenge. This is a standard task in computer vision, where models try to classify an image into 1000 classes, like "Zebra" or "Dishwasher". Compared with previous DNN models, InceptionV3 has one of the most complex network architectures in computer vision.

The service is composed of two services: an InceptionV3 network that output a vector representing the recognized image class, and an decoding service for Image Net that translates the previous vector to human-readable format. These two services are sequentially connected. By using Zoo, we can deploy this new services to local Raspberry Pi devices with only one line of command. With minor change of configuration, the service can also be deployed to our server on Digital Ocean, a cloud infrastructure provider. Currently we have used this localized deployment instance for internal automatic image tagging tasks. For the latter instance, we use it to construct an online demo of this service.

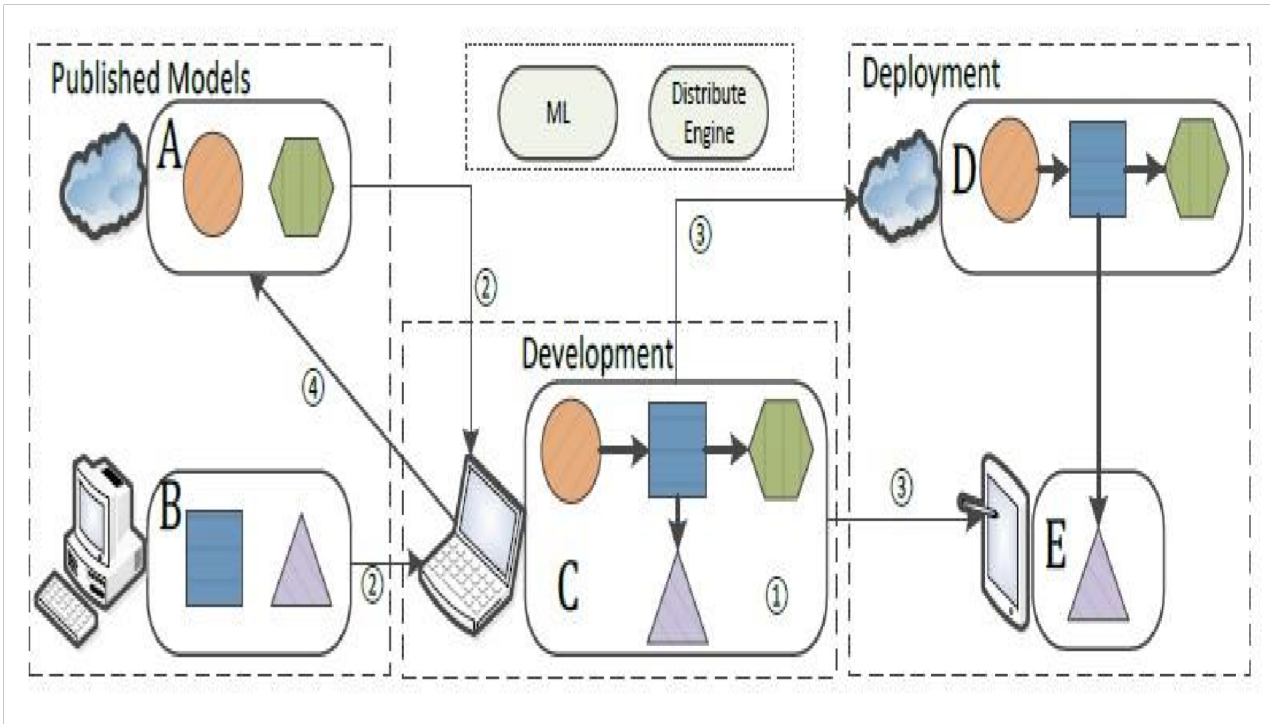

Figure 3.1 : Zoo system architecture

CHAPTER 4

PERFORMANCE EVALUATION

In this section we present preliminary experiment results with Zoo system. We want to answer two questions here First, we choose to use Owl to base the Zoo system on. To what degree does its expressiveness has shown? Does it bring any performance trade-off? Second, we deploy services on local devices, so what are their performance compared with popular cloud-based analytic solutions such as Google ML API?

Let's start with the first question. As stated before, Owl provides extraordinary expressiveness and flexibility. One example to show this is that using Owl to constructs the InceptionV3 network only takes 150 lines of code (LoC), compared with the 400 or more LoC used by Tensorflow. Another one is that, we insert instrumentation code into Owl to enable collecting forward computation latency of each node(or layer according to some literature, et pass.) in a neural network when doing inference. Adding this feature only takes 50 LoC in the source code of Owl.

Since the Zoo relies on inference using Owl's Neural Network module, we want to compare the inference time on Owl and the other state-of-art deep learning platforms. We choose three representative DNN models that vary greatly in both architecture complexity and parameter sizes:

1. one small neural network (LeNet-5 (LeCun et al. 1998)) that only consists of 8 nodes and contains about 240KB parameters (each parameter in a model is represented by a 32bitfloat number, et pass.) for the MNIST handwriting recognition task;
2. A VGG16 (Simonyan and Zisserman 2014)model that has a simple architecture with 38 nodes but a large amount of parameters (500MB) for real-world image recognition task;
3. An InceptionV3 model also for image recognition, with less parameters (100MB), but a far more complex architecture (313 nodes). We compare the time it takes for each model to finish its inference task on different platforms: Owl, TensorFlow, and Caffe2. Each measurement is repeated 20 times.

The results are shown in Figure 4.1. Regardless of great diversities in these models' architectures and sizes, Owl takes less time to do inference than Tensorflow and Caffe2. It means

that Owl can achieve both expressiveness and good performance. The superior performance of Owl on large models is attributed to its efficient math operations.

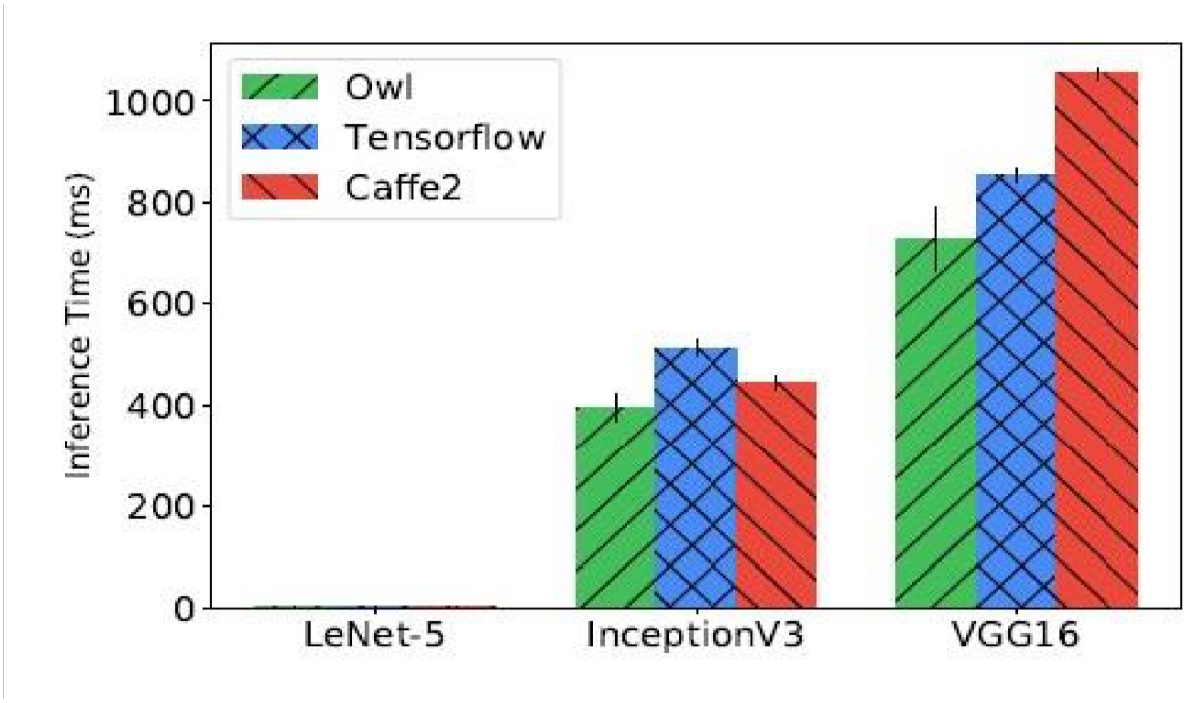

Figure 4.1: Inference time comparison on both the platforms

Next, we investigate the performance of Zoo system compared with Google ML API. The Google Cloud Vision API (Google 2017a) encapsulates machine learning models in a REST API. It can classify images into thousands of categories as well as detects individual objects and faces within images, and finds and reads printed words contained within images. Its workflow is simple: a user creates service token on the Google Cloud Platform (GCP), and passes the token and an image to Google's server for processing, and then the processed results will be returned to the user as response in the form of JSON. In this evaluation we compare the image classification development.

We deploy Vision API service on GCP. The network connection bandwidth is 34 Mbps measured onspeedtest.net. In the previous section we have shown a deployment example of our image classification service on local devices with Zoo. For this evaluation, we deploy this service on a Think pad T460s laptop with a Intel Core i5-6200U CPU. First, we have collected 100 animal

images as a dataset2. These images are of different sizes, ranging from7KB to 1243KB, to better simulate users' requests in real world. Specifically, we compare the time required for both method required for both method to process different number of images.

The results are shown in Figure 4.2. It has shown that our Zoo service achieves lower response latency. When the number of input images from a user gradually increases from 5images to 25 images, the response time of the Zoo service increases linearly, which means the process time of each image basically keeps constant despite the size difference of input images, while the increase speed Google service's response time seems to grow with more input images. The measurement at each point is repeated for 10 times. Note that the Zoo service response time keep stable (about 0.6sper image) with very small deviation, and thus predictable. The same cannot be said of Google Vision service. It shows relatively large deviation, and with the change of network connection, the response time could easily fluctuate and become unpredictable.

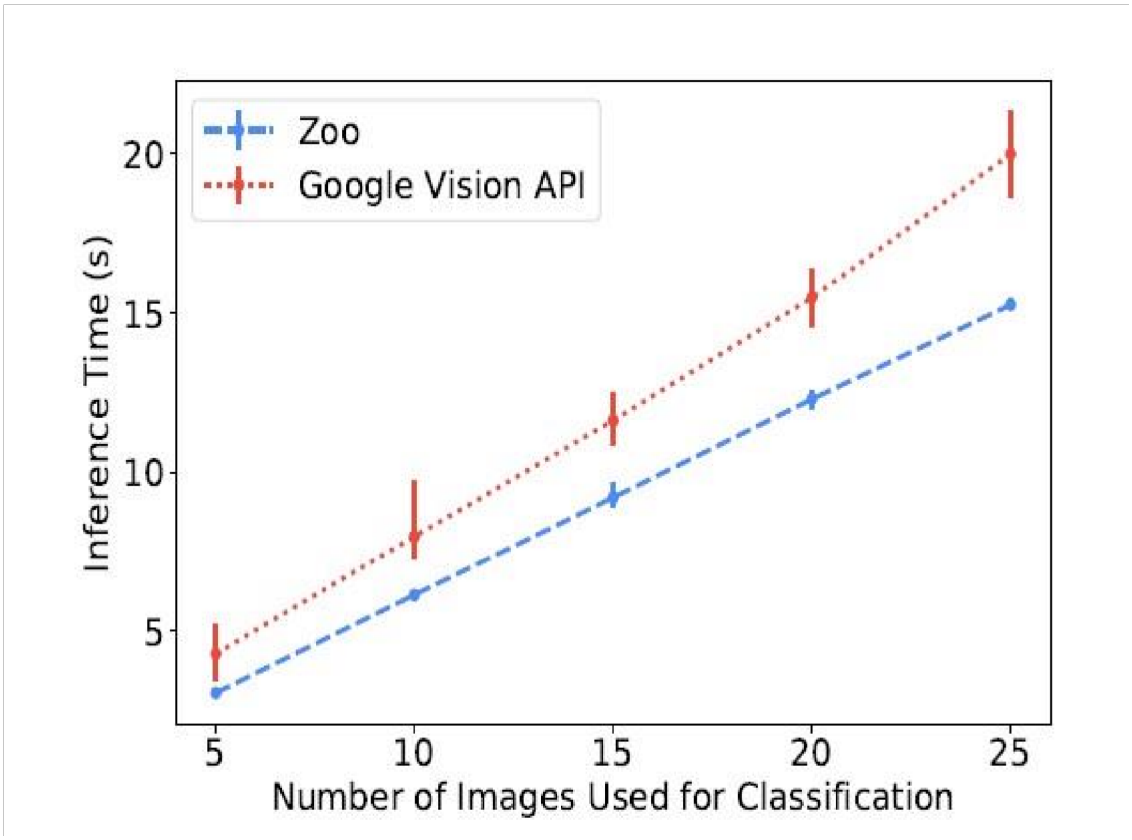

Figure 4.2: Performance comparison of local image classification services deployed Zoo with Google's Machine Learning Vision service.

CONCLUSION

Machine Learning based data analytics have been widely adopted in today's online applications and services. However, with social awareness of privacy and personal data rapidly rising, they also bring about a pressing societal issue: data privacy. Most existing solutions to this issue focus on enhancing the currently popular cloud-based analytics approach, where users' data need to be collected to central server for processing. Besides data privacy, this approach is also prone to issues such as increased service response latency, communication cost, and single point failure.

In this paper, I argue that to avoid those costs, reduce latency in data processing, and minimize the raw data revealed to service providers, many future AI and ML services could be deployed on users' devices at the Internet edge rather than putting everything on the cloud. This brings many challenges. Besides the widely discussed factor of resource limitation on edge devices, I identify two other challenges that are not yet recognized in existing literature: lack of suitable models for users, and difficulties in deploying services for users. I present the design of our Zoo system to support the construction, compose, and easy deployment of ML models on edge and local devices. Then I show how services can be composed and deployed with Zoo, and its superior performance compared with existing deep learning platform and cloud-based services. I believe this area of research is only just beginning to gain momentum.

BIBILOGRAPHY

[1]. [Babuji et al. 2016] Babuji, Y. N.; Chard, K.; Gerow, A.; and Duede, E. 2016. Cloud kotta: Enabling secure and scalable data analytics in the cloud. In Big Data (Big Data), 2016 IEEE International Conference on, 302–310. IEEE.

[2]. [European-Commission 2016] European-Commission.2016. Protection of personal data. http://ec.europa.eu/justice/data-protection/reform/index_en.html. Accessed: 2017- 10- 07.

[3]. [Google 2017b] Google. 2017b. Google privacy. https://privacy.google.com. Accessed Nov.

11, 2017.

[4]. [European-Commission 2016] European-Commission.2016. Protection of personal data. http://ec.europa.eu/justice/data-protection.

[5]. [Meeker 2016] Meeker, M. 2016. Internet trends 2016 report. http://www.kpcb.com/internettrends. [6]. [Fredrikson, Jha, and Risten part 2015] Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22ndACM SIGSAC Conference on Computer and Communications Security, 1322–1333.

ACM.

[7]. [Wang 2017] Wang, L. 2017. Owl: A general-purpose numerical library in ocaml. CoRR abs/1707.09616.