Privacy in Deep Learning

A Project Report

submitted by

KAKARLA HARISH REDDY

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY AND MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2019

THESIS CERTIFICATE

This is to certify that the thesis titled **Privacy in Deep Learning**, submitted by **Kakarla Harish Reddy**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Chester Rebeiro
Research Guide
Assistant Professor
Dept.of Computer Science and Engineering
IIT-Madras, 600 036

Prof. Janakiraman Viraraghavan
Research Co-Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: IIT Madras, Chennai

Date: 5th May 2019

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Professor Chester Rebeiro and Professor Pratyush Kumar, my research supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work.

I would also like to thank Patanjali SLPSK, for his valuable input and assistance in keeping my progress on schedule.

ABSTRACT

KEYWORDS: Deep Learning, User Privacy, Split NN

When it comes to applying deep learning to a problem which involves medical, financial, or other types of sensitive data, it is not enough to have accurate predictions, one must pay careful attention in maintaining data privacy and security. Legal and ethical requirements may prevent the use of cloud-based deep learning solutions for such tasks. In this work we propose a framework where the user rely on cloud-based services to achieve the results for his deep learning models. Since the definition of privacy is task dependent, we define what is means for the user data to be private, and accordingly define the goal of the adversary in the current situation. We also classify the adversary based on the information he has and what he is capable of. The efficiency and effectiveness of our framework are demonstrated with CIFAR-10 dataset.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DLaaS	Deep Learning as a Service
SRGAN	Super Resolution Generative Adversarial Network

NOTATION

μ	Mean
σ	Sigma
$N(\mu, \sigma)$	Normal distribution with mean = μ and sigma = σ
λ_1, λ_2	Hyperparameters used is the loss function

CHAPTER 1

INTRODUCTION

In recent times, the progress in neural networks enabled us to achieve remarkable results in a wide range of applications including image classification, speech recognition, object detection, etc. These advances were enabled, partly because of the availability of large representative datasets for training neural networks. Often these datasets may contain sensitive information. When applying deep learning to a problem which involves medical, financial, or other types of sensitive data, it's not enough if the predictions are accurate, we should also take into consideration the privacy and security aspects of the data involved.

One way to guarantee the privacy of user data is to perform the training and inference of neural network in a local trusted machine. We cannot expect everyone to have access to resources which can accomplish this task. This problem can be solved by using a third party to bear the computation load. This idea is known as **Deep Learning as a Service**(DLaaS). When using DLaaS one has to be mindful about the privacy of user data.

The notion of privacy can be discussed during the training phase and the inference phase. To get a notion of training the neural network in such a setting refer *privacy-preserving data-mining* [1]. One possible solution to training while preserving privacy lies in the concept of *differential privacy* [2]. The current work focuses on preserving the privacy of user data during the inference phase.

We assume that a trained model is already loaded on to the server and we discuss the steps the user has to take to protect his data against the adversary. In these models, the notion of privacy is task specific. Hence we define the notion of privacy for the current task at hand. We also analyze the capabilities of adversary based on the amount of data he can access.

CHAPTER 2

PREVIOUS WORKS

Let's take a look at some of the previous approaches used to solve this problem and discuss the pros and cons of some of the corresponding works.

Approaches	***Works***
Homomorphic Encryption (HE)	**Cryptonets**[3], CryptoDL[4]
Garbled Circuits (GC)	**DeepSecure**[5]
Additive Secret Sharing (A-SS)	Sharemind[6]
Hybrid (Mixture of various approaches)	**Chameleon** [7], SecureML[8], MiniONN[9]

Table 2.1: Previous works

2.1 CryptoNets

This framework allows a data owner to send their data in an encrypted form to a cloud service that hosts the network. The encryption ensures that the data remains confidential since the cloud does not have access to the keys needed to decrypt it. Nevertheless, the cloud service is capable of applying the neural network to the encrypted data to make encrypted predictions, and also return them in encrypted form. These encrypted predictions can be sent back to the owner of the secret key who can decrypt them. Therefore, the cloud service does not gain any information about the raw data nor about the prediction it made.

This cryptosystem allows computing the polynomial functions of a fixed maximal degree on the encrypted data. High degree polynomial computation requires the use of large parameters in the scheme, which results in larger encrypted messages and slower computation times. Hence, a primary task in making practical use of this system is to present the desired computation as a low-degree polynomial.

The training phase requires non-linear activation functions, but evaluation phase needs low-degree polynomial functions. Hence non-linear low degree polynomial functions are incorporated in training phase also. This will result in slight decline in accuracy which can be considered as a fair tradeoff to achieve privacy. For the sake of time-efficient evaluation, consecutive layers that use only linear transformations, such as the weighted-sum or mean pooling, can be collapsed.

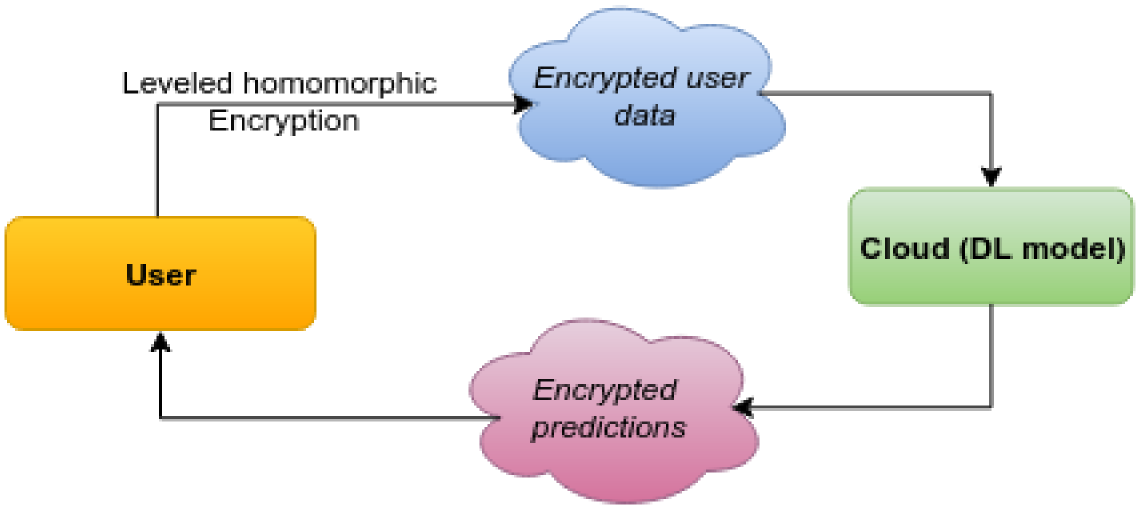


Figure 2.1: CryptoNets framework

2.2 DeepSecure

This framework enables scalable execution of the state-of-the-art Deep Learning (DL) models in a privacy-preserving setting. DeepSecure targets scenarios in which neither of the involved parties including the cloud servers that hold the DL model parameters or the delegating clients who own the data is willing to reveal their information. The secure DL computation in DeepSecure is performed using **Yao's Garbled Circuit**(GC) [10] protocol. This work performs GC-optimized realization of various components used in DL.

DeepSecure enables computing the pertinent data inference label in a provably-secure setting while keeping both the DL model's parameters and data sample private. To perform a particular data inference, the netlist of the publicly known DL architecture should be generated prior to the execution of the GC protocol. The execution of the GC protocol involves four main steps

1. The client (data owner) garbles the Boolean circuit of the DL architecture.
2. The client sends the computed garbled tables from the first step to the cloud server along with her input wire labels. Both client and the cloud server then engage in a **1-out-of-2 Oblivious Transfer** (OT) [11] protocol to obliviously transfer the wire labels associated with cloud server's inputs.
3. The cloud server evaluates (executes) the garbled circuit and computes the corresponding encrypted data inference.
4. The encrypted result is sent back to the client to be decrypted using the garbled keys so that the true inference label is revealed.

2.3 Chameleon

The main design goal behind Chameleon is to create a framework that combines the advantages of the previous secure computation methodologies. This is a hybrid (mixed-protocol) framework for **secure function evaluation**(SFE) which enables two parties to jointly compute a function without disclosing their private inputs. Chameleon combines the best aspects of generic SFE protocols with the ones that are based upon additive secret sharing.

In particular, the framework performs linear operations using the additively secret shared values and nonlinear operations using Yao's Garbled Circuits. Chameleon departs from the common assumption of additive or linear secret sharing models where three or more parties need to communicate in the online phase: the framework allows two parties with private inputs to communicate in the online phase under the assumption of a third node generating correlated randomness in an offline phase. Almost all of the heavy cryptographic operations are precomputed in an offline phase which substantially reduces the communication overhead.

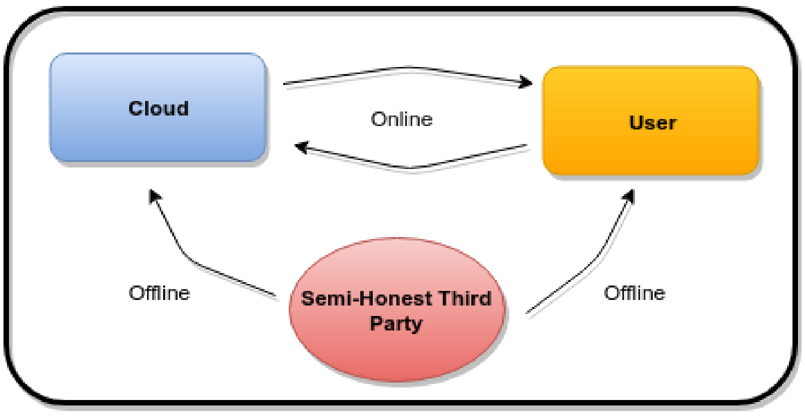


Figure 2.2: Chameleon framework

Pros	***Cons***
Cryptonets	
Can encode n values into a single polynomial, operate on this polynomial and decode the n different results	No support for floating-point numbers
The user need not be online during the evaluation phase	Noise growth is strong in multiplication => Bigger the network slower the computation
Well suitable for batch settings	Have to retrain the network with polynomial activation functions => trade off between accuracy and privacy
DeepSecure	
This method does not have privacy/accuracy trade-off	Have to retrain the network after mapping training images into sparse subspace - data preprocessing
Well suitable for streaming settings	User has to be online during the evaluation phase
Non-linear activation functions can be accurately represented by Boolean circuits	DL network is pruned prior to netlist generation
Chameleon	
Less computation and communication overhead	Involves a Semi-honest Third Party
	User has to be online during the evaluation phase

Table 2.2: Pros and Cons of previous works

CHAPTER 3

PROPOSED FRAMEWORK

The goal of this work is to come up with a framework that protects the privacy of user data in a DLaaS setting and at the same time make sure it is time and compute efficient. To this end, we propose a **Split NN framework** where a NN split into two parts, the first part (M1) resides at the user end, while the second part (M2) resides at the server end. Ideally speaking the split is made in such a way that less computation is performed at the user end and more computation is performed at the server end. Now instead of sending the features directly from M1 to M2, the intermediate features are passed through a **perturbation layer**(P). The purpose of P is to perturb the features before sending them to the server. The functionality of this layer depends on various factors which are discussed in further sections.

In this work the focus is on one particular task, **Image Classification**. The user has a set of images, and a pre-trained model to classify these images. When he uses our proposed framework for inference, the process goes like this:

1. Split the NN into M1 and M2
2. Load M2 on to the server.
3. Pass the image through M1 and get the intermediate features
4. Pass the intermediate features through P and get the perturbed features
5. Pass the perturbed features to M2 and get the class label as output.

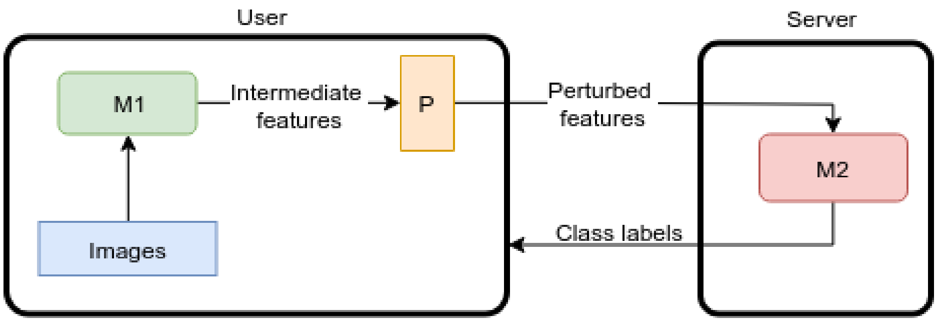


Figure 3.1: Proposed Framework

CHAPTER 4

ADVERSARY

4.1 Adversarial Model

Now that we have defined our framework it is time to define the adversarial model we are considering for the current setting.

A reasonable definition of privacy would be, '**the ability of an individual or group to seclude themselves, or information about themselves, and thereby express themselves selectively**'. Even though we have a general idea of privacy, when it comes to defining privacy in the current context, we have to place some constraints when defining what it means to keep the user data private.

For example,

1. If the data-set at hand was obtained by conducting a survey, the people participated in the survey and the answers they gave need not be private individually, but the relation between them (**who answered what?** should be kept private).
2. If the data-set at hand is a set of images, then all the images in the data-set have to kept private if they might reveal something about the user.

On these lines, we say that the notion of privacy is dependent on the **task at hand** and the **constraints posed by the user**.

Here we consider the case where the user wants to keep all the images he uses for classification to be private. Hence we define our adversary as an entity who wishes to obtain these images. In this work we focus on the **network adversary**. We assume that the adversary can eavesdrop on the communication channel between the user and the server and gain access to the data beign transferred.

Since the user is sending the output of P to M2 rather than sending the images themselves, obtaining the images used for classification is not a straightforward task. The adversary has to reconstruct the images from the intermediate features. For this image reconstruction task, we use the state-of-the-art generative NN architecture based

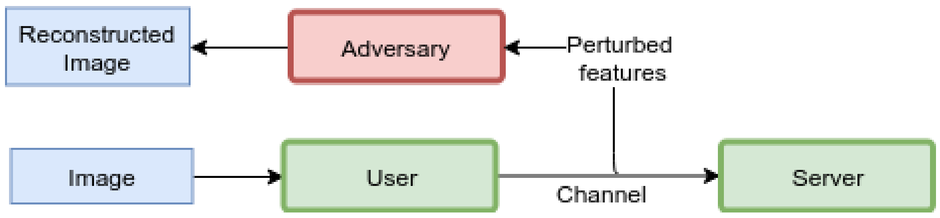


Figure 4.1: Adversarial Model

on ResNet blocks [12], which has demonstrated good performance for different image recovery tasks, including super-resolution [13], denoising autoencoder [14]. Refer appendix for the exact architecture of the Super Resolution Generative Adversarial Network (SRGAN) used for this work.

Given the adversary knows the situation he is trying to tackle, he knows the distribution of the data that is being used. For example, if the adversary knows that he is targeting a hospital then he can take a guess at what kind of images are being used (X-rays, MRI scans, etc). To consider the worst case scenario let us assume that the adversary knows the training set that has been used to train our model. Using this data-set the adversary trains his SRGAN to accept the intermediate features and reconstruct the image.

4.2 How powerful is the chosen adversary?

To get an idea of how well the reconstructed images are, we pass these images through a classifier and measure the accuracy. From here on, we term this quantity as **SRGAN Accuracy**. Higher SRGAN Accuracy implies more powerful adversary. In this work we used VGG16 as the image classifier which the user splits across himself and the server. We used CIFAR-10 data to train it. The reconstruction results when there is no perturbation layer (P) are shown in figure 4.2 .

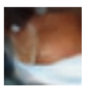
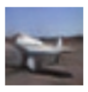
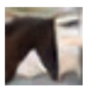
Test images (Accuracy = 92%)							
Reconstructed images (Accuracy = 76%)							

Figure 4.2: Reconstructed images when there is no perturbation layer

We further claim that the adversary need not have access to all the intermediate features of the images in the training dataset to perform this well. Let's have a look at how the adversary performs if he has access to limited amount of data.

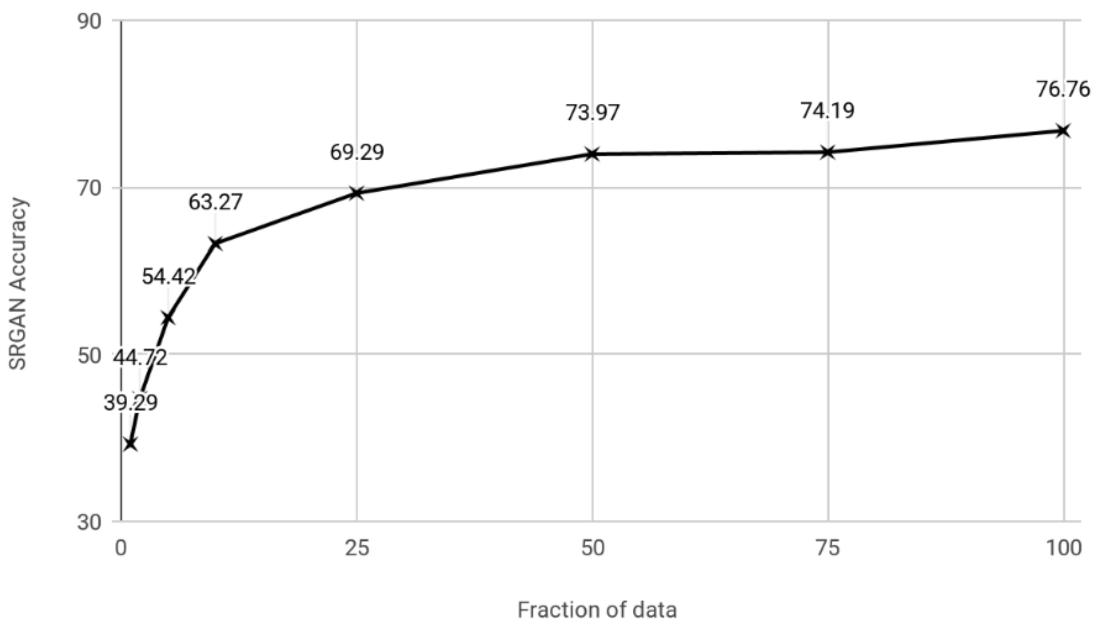


Figure 4.3: SRGAN Accuracy (vs) Fraction of data

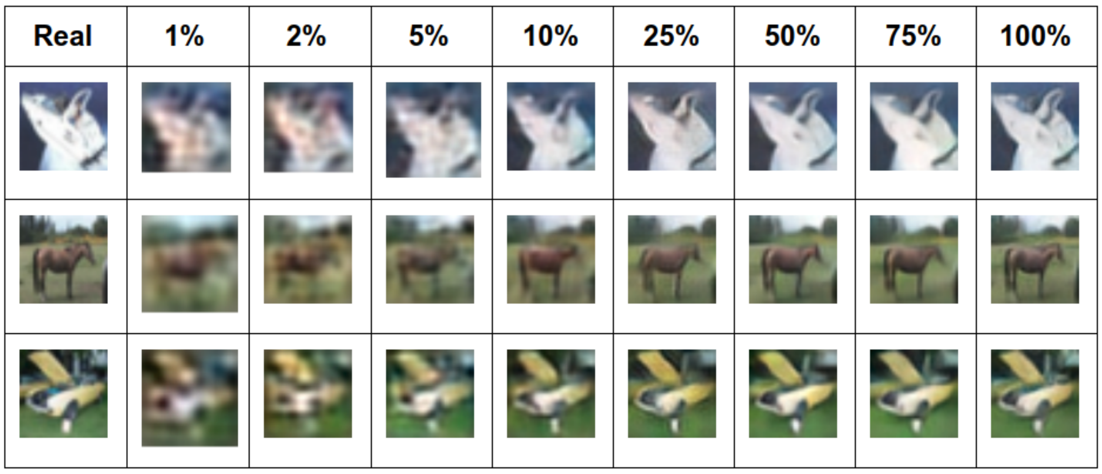


Figure 4.4: Reconstructed images when the adversary has access to partial data

Just by getting access to **25%** of the data the adversary is able to achieve **~70%** SR-GAN Accuracy. From this, we conclude that we are considering a remarkably powerful adversary.

CHAPTER 5

CLASSIFICATION OF ADVERSARIAL MODELS

As the notion of privacy varies across tasks and the constraints posed by the user, before proposing our defensive mechanism (action performed by perturbation layer), we classify the adversary based on the information he has and what he is capable of. The classification is as follows:

Naive Adversary (A1):

Is oblivious to the fact that a defensive mechanism is being deployed.

Strong Adversary (A2):

Knows that a defensive mechanism is being deployed, but has no idea what it is.

Best Adversary (A3):

Knows the defensive mechanism that is being deployed.

To get a better understanding of this classification, we consider one particular defensive mechanism and observe how each adversary performs. Consider the perturbation layer in this case to be **Noise layer**.

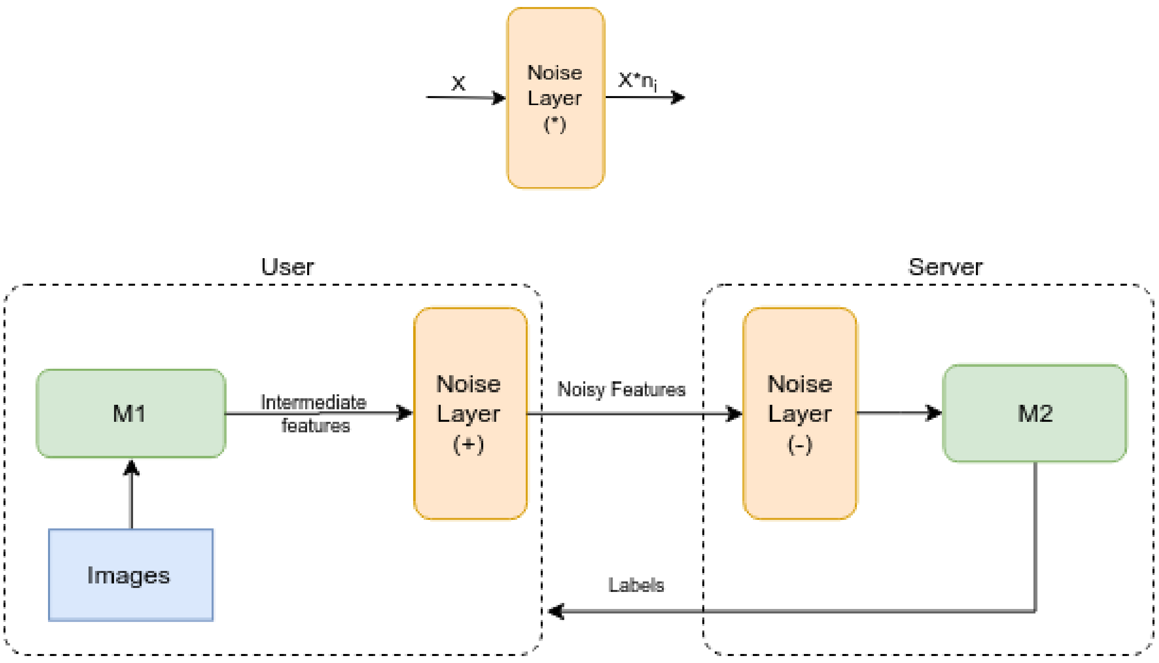


Figure 5.1: Framework for comparing adversaries

The user adds noise sampled from a Normal distribution ***N***(μ,σ) to the output of M1 and sends the noisy features to server. For this particular example assume that the server also knows this (μ, σ). Once the noisy features are sent to the server it then subtracts the noise sampled from the same distribution $\boldsymbol{N}(\mu, \sigma)$ and performs the classification. We vary (μ, σ) and observe how the user is performing.

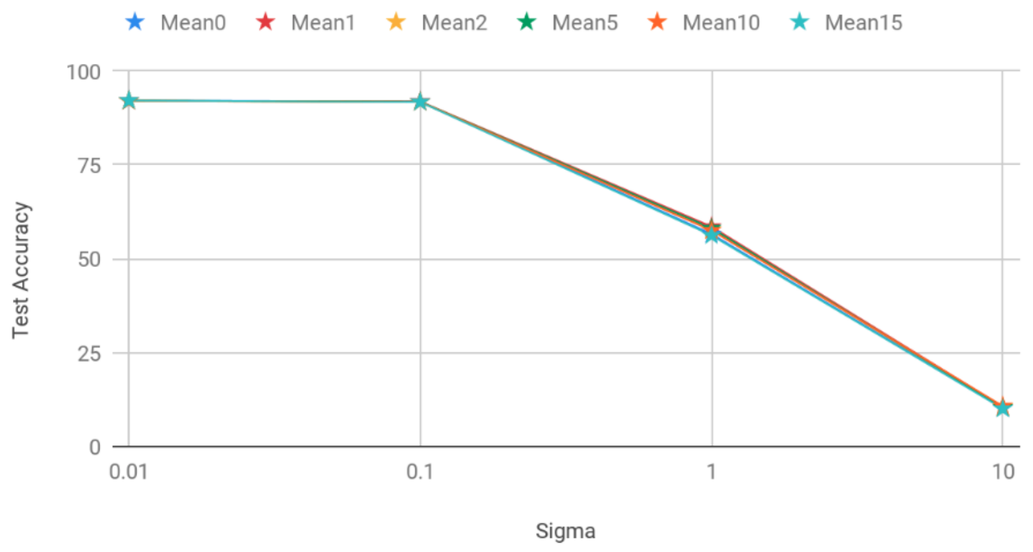


Figure 5.2: Test Accuracy of pre-trained VGG16 with noise layer

The above results are as expected. Since we are subtracting the noise sampled from the same distribution, as long as the noise is sampled from a low variate distribution we can remove it completely. As the variance increase the samples we pick will become more dispersed which will decrease the Test Accuracy.

Now let us observe how each adversary is performing. For comparison, here we only consider the cases when the noise is sampled from ***N****(0,0.01)*, ***N****(0,1.0)*, ***N****(5,0.01)*, ***N****(5,1.0)*. The results for the remaining distributions can be found in chapter 7.

Naive Adversary (A1): Since A1 does not know that a defensive mechanism is being deployed, he uses the noisy features directly to reconstruct the images.

Stronger Adversary (A2): Since A2 knows that a defensive mechanism is being deployed, he knows that the features he accessed are noisy and hence retrains his SR-GAN and then tries to reconstruct the images.

Best Adversary (A3): Since A3 is aware that the defensive mechanism being deployed is additive noise, he normalizes the features and then trains the SRGAN to perform the reconstruction.

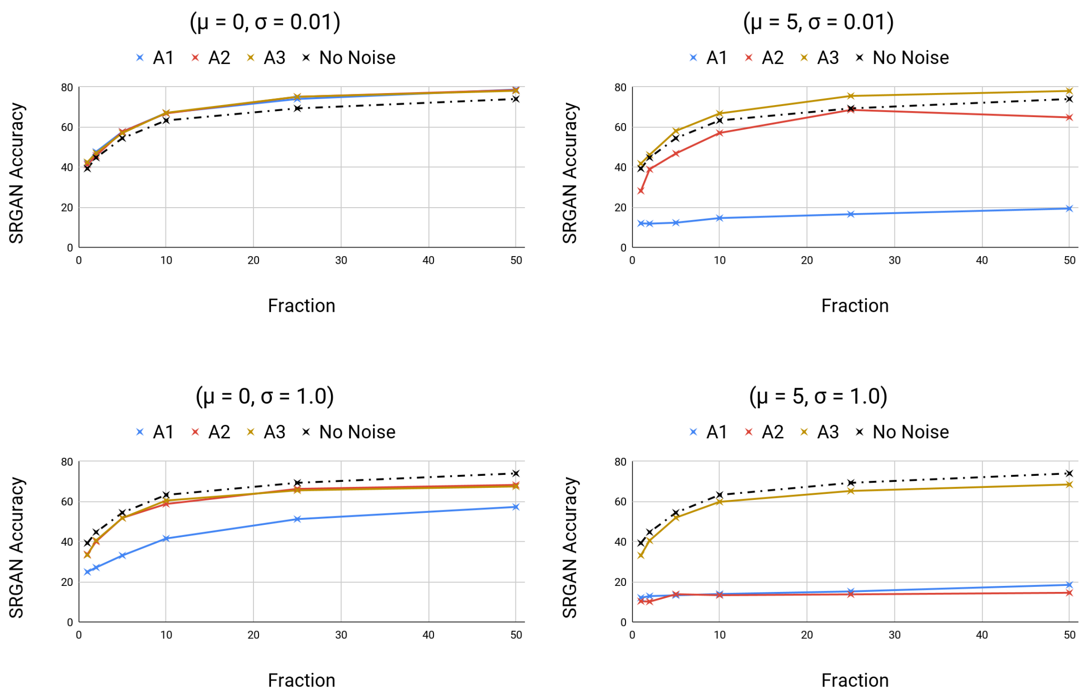


Figure 5.3: Comparing A1, A2, A3

By comparing the results we find that,

- A1 performs poorly as μ or σ increases.
- A2 performs relatively well compared to A1, for high μ and σ.
- A3 performs well in all the cases.

CHAPTER 6

PROPOSED DEFENSIVE MECHANISM

In the current context we use the SRGAN Accuracy (i.e. classification accuracy of the images reconstructed by the adversary) as our privacy metric. High SRGAN Accuracy implies less privacy. Our goal is to minimize this quantity without compromising on the Test Accuracy (i.e. classification accuracy of the user data-set).

X - Test Accuracy when no defensive mechanism is applied

Y - Test Accuracy when defensive mechanism is applied

Z - SRGAN Accuracy when defensive mechanism is applied

Objective: *maximize Y-Z* and *minimize X-Y*

From the discussion in the previous chapter, it is clear that A3 is a more powerful adversary when compared to A2 and A1. If we propose a defensive mechanism which works against A3, we don't have to worry about the other two. Hence from here on we consider A3 as our adversary.

Consider the single split model discussed in the previous chapter. The idea was to pass the intermediate features through a noise layer while training and control the amount of noise being added to find a trade-off between Y and Z.

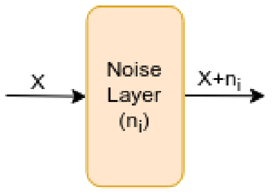


Figure 6.1: Noise Layer

To control the magnitude of the noise added, we modified the loss function as,

$$\boxed{L' = L - \lambda_1 * (\Sigma|n_i|) + \lambda_2 * (\Sigma n_i^2)}$$

n_i - parameters of the noise layer λ_1, λ_2 - hyperparameters

As the objective of the NN during training is to reduce the loss, λ_1 tries to increase n_i but λ_2 tries to decrease it. Hence the final magnitude of the noise added will depend on the ratio λ_1/λ_2. The problem with this idea is the **BatchNorm layer** present in the network. This layer normalizes the input before propagating it to the next layer. So the noise added will not propagate to the last layer. Hence the Test Accuracy does not take a hit, and SRGAN Accuracy will be high as the adversary normalizes the features before using them.

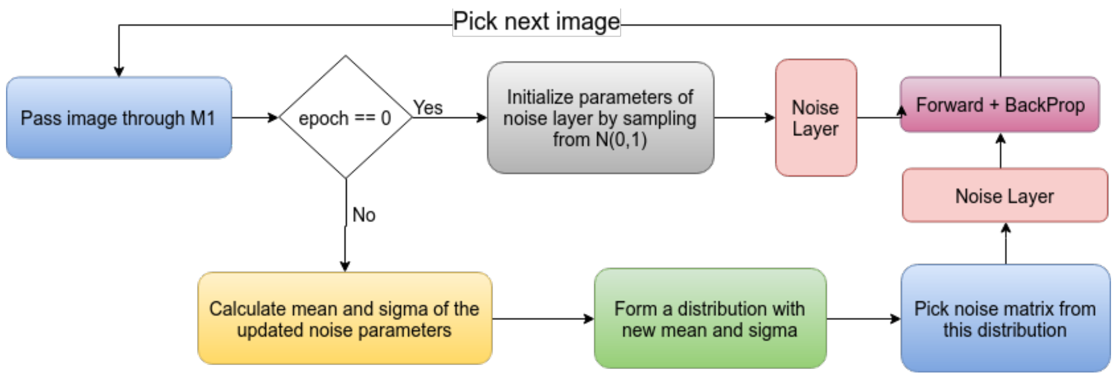
This result motivates that the magnitude of the noise being added does not play a crucial role in deciding the accuracy (both for the user and the adversary). Hence we focus on the variance of the noise. We increase the variance of the noise being added in a controlled fashion while training the model itself so that the trained model is robust enough to account for this noise but the adversary fails to reconstruct the images. The training process goes like this,

```
Pass the image through M1
If (epoch == 0):
    Initialize the parameters of the noise layer by sampling them from N(0,1)
If (epoch != 0):
    Form a normal distribution with the $(\mu, \sigma)$ of the updated parameters of the Noise layer
    Sample the parameters of the noise layer from this N$(\mu, \sigma)$
Pass the output of M1 to Noise layer
Perform forward and backward prop
Update the parameters of the Noise layer
If (epoch == last epoch):
    exit
Repeat
```

To control the variance of the noise added, we modify the loss function as,

$$\boxed{L' = L - \lambda_1 * (\sigma) + \lambda_2 * (\sigma^2)}$$

We start off with **N**(0,1). As the goal of the NN during training is to minimize the loss(L'), λ_1 tries to increase the variance of the noise being added and λ_2 tries to



Pick next image
Pass image through M1
epoch == 0
Yes
No
Initialize parameters of noise layer by sampling from N(0,1)
Noise Layer
Forward + BackProp
Noise Layer
Calculate mean and sigma of the updated noise parameters
Form a distribution with new mean and sigma
Pick noise matrix from this distribution

Figure 6.2: Training process for proposed defensive mechanism

decrease it. So the ratio of λ_1/λ_2 decides the variance of the noise being added finally. Higher the ratio of λ_1/λ_2 higher the variance. Each time we train the network, it learns to fit a different hyperplane and hence our accuracy varies across different trails. This variation in accuracy is proportional to the ratio of σ which depends on λ_1/λ_2.

Now based on the situation in which our framework is used and the constrains placed by the user we can tune λ_1 and λ_2 to get the desired results. As the server is oblivious to the values of λ_1 and λ_2, this mechanism can also be implemented in situations where the user does not trust the server.

Let's compare the performance of user and the adversary for $\lambda_1/\lambda_2 = 25$.

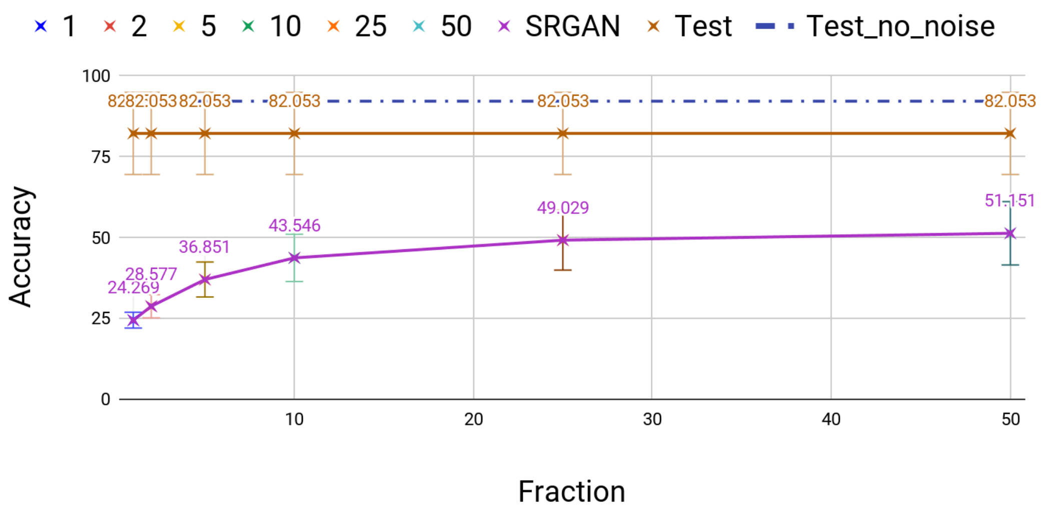


Figure 6.3: Comparing the performance of user and the adversary for $\lambda_1/\lambda_2 = 25$

We can see a significant difference in Test Accuracy (user) and SRGAN Accuracy (adversary). The results for different values of λ_1/λ_2 can be found in chapter 7.

CHAPTER 7

EXPERIMENTS AND RESULTS

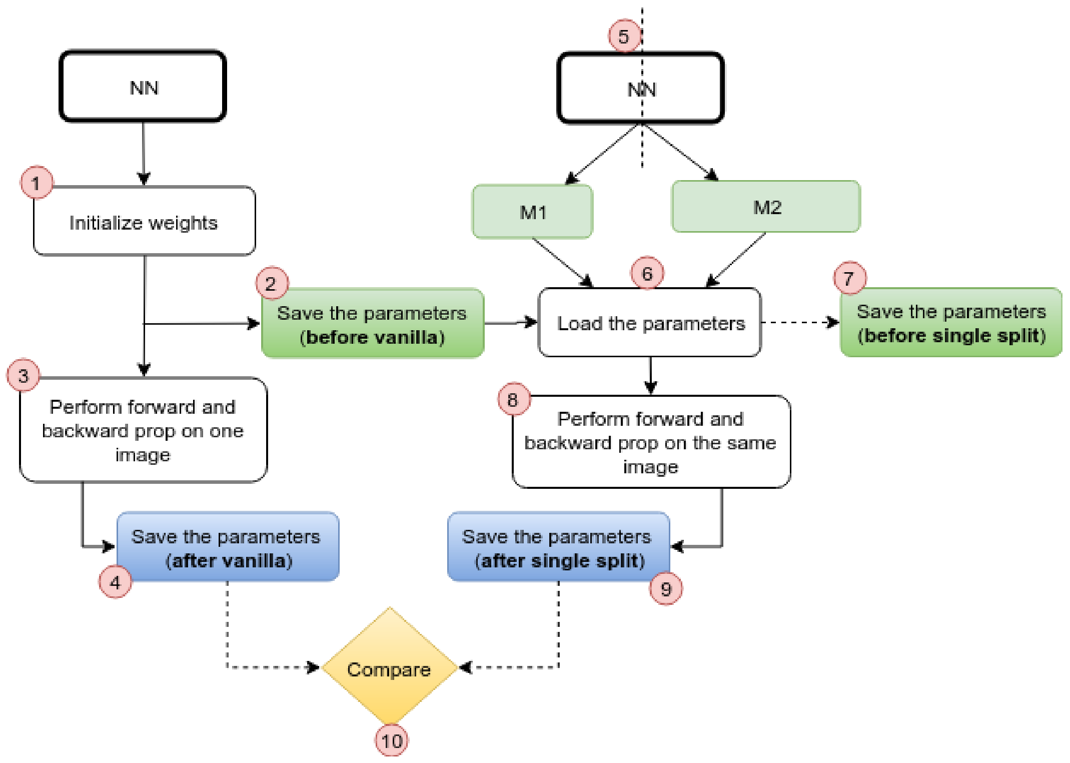
Data set $\rightarrow CIFAR-10$	**Framework** $\rightarrow PyTorch$
User $\rightarrow VGG16$	**Adversary** $\rightarrow SRGAN$

7.1 Validation of the Single Split model

This experiment validates the proposed single split neural network model for training.

- Neural Network before performing the split is termed as Vanilla NN.
- Neural Network after performing the split is termed as Single Split NN.

1. *Initialize the parameters of Vanilla NN*
2. *Save the parameters of the network(**Params before for Vanilla NN**)*
3. *Perform forward prop and backward prop for one image*
4. *Save the parameters of the network(**Params after for Vanilla NN**)*
5. *Split the NN into two parts M1 and M2*
6. *Load **Params before for Vanilla NN** into Single Split NN*
7. *Save the parameters of the network(**Params before for Single Split NN**)*
8. *Perform forward and backward prop for the same image*
9. *Save the parameters of the network(**Params after for Single Split NN**)*
10. *Compare the results from steps 4 and 9*



5
NN
1
Initialize weights
M1
M2
2
Save the parameters (before vanilla)
6
Load the parameters
7
Save the parameters (before single split)
3
Perform forward and backward prop on one image
8
Perform forward and backward prop on the same image
Save the parameters (after vanilla)
4
Save the parameters (after single split)
9
Compare
10

Figure 7.1: Validation of Single Split

7.2 Comparing A1, A2, A3

The performance of A1, A2 and A3 is compared for different $\mathbf{N}(\mu, \sigma)$.

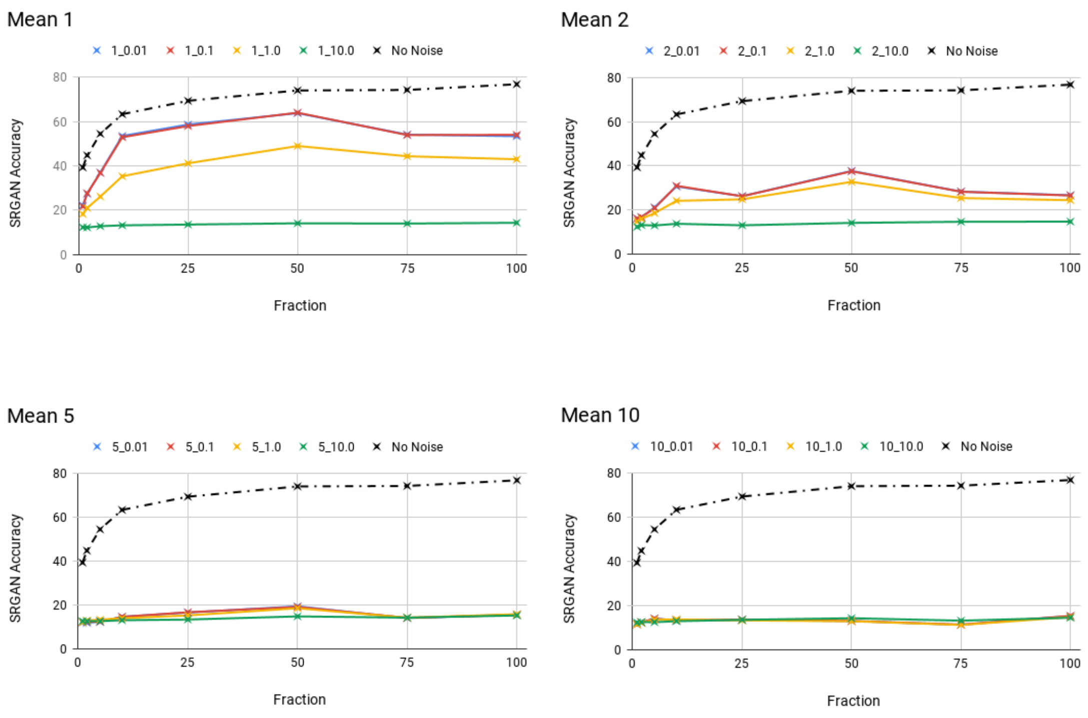
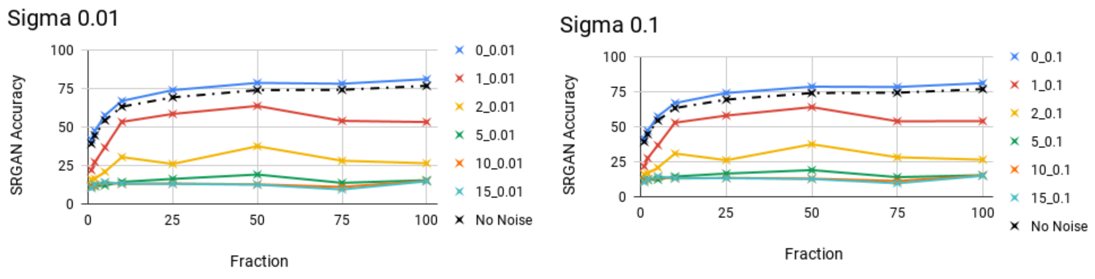


Figure 7.2: Performance of A1 for fixed μ and varying σ



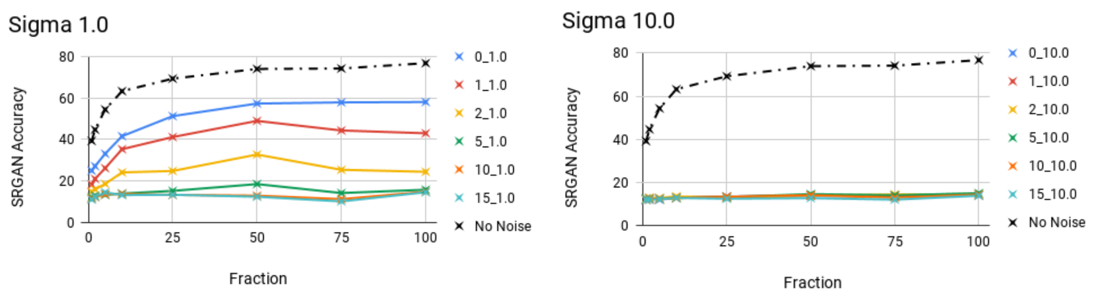


Figure 7.3: Performance of A1 for fixed σ and varying μ

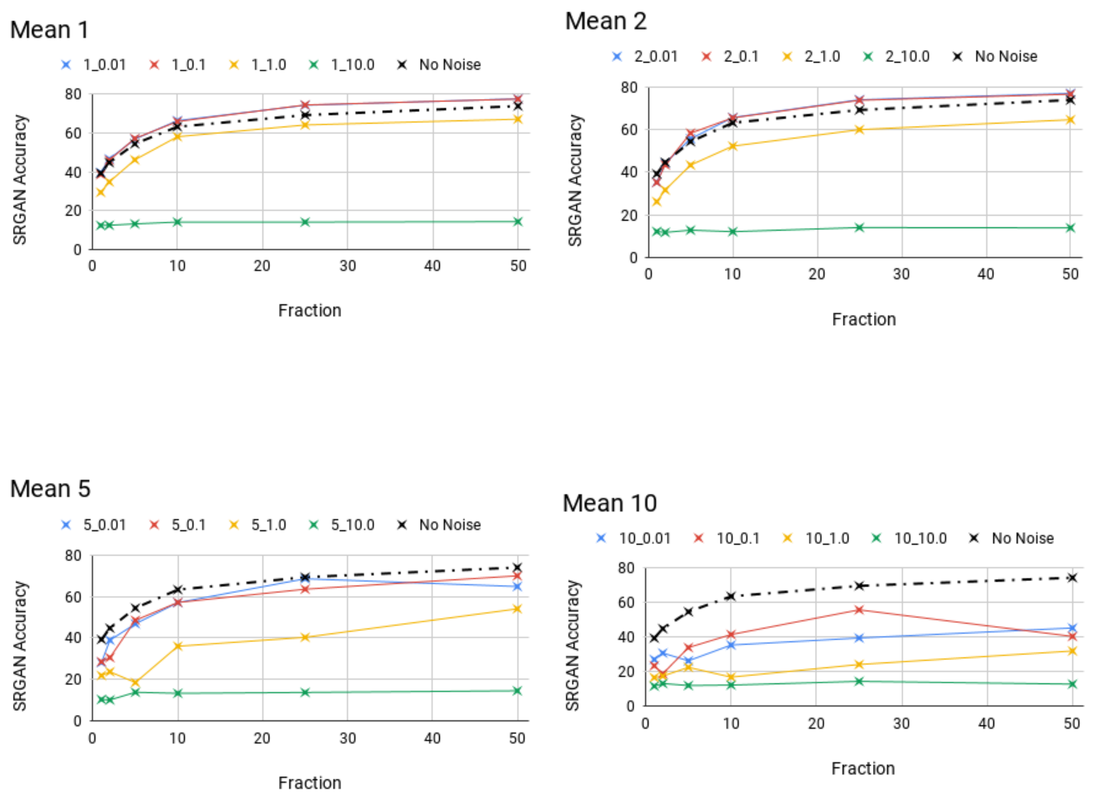
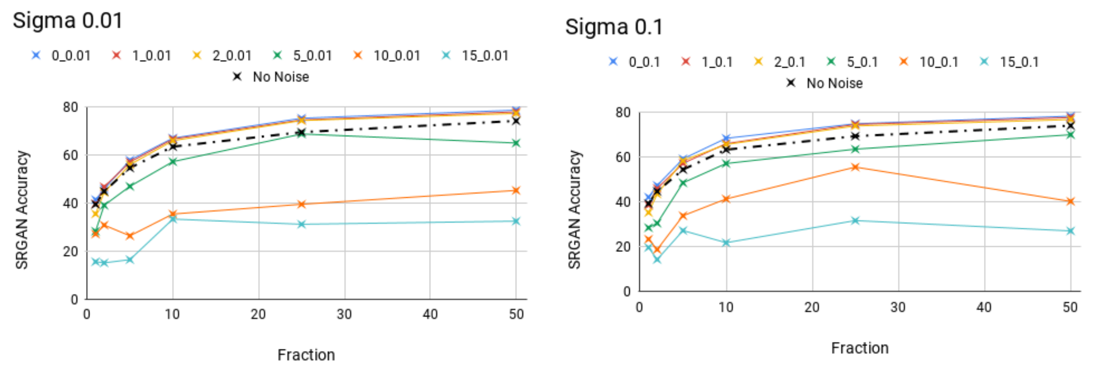


Figure 7.4: Performance of A2 for fixed μ and varying σ



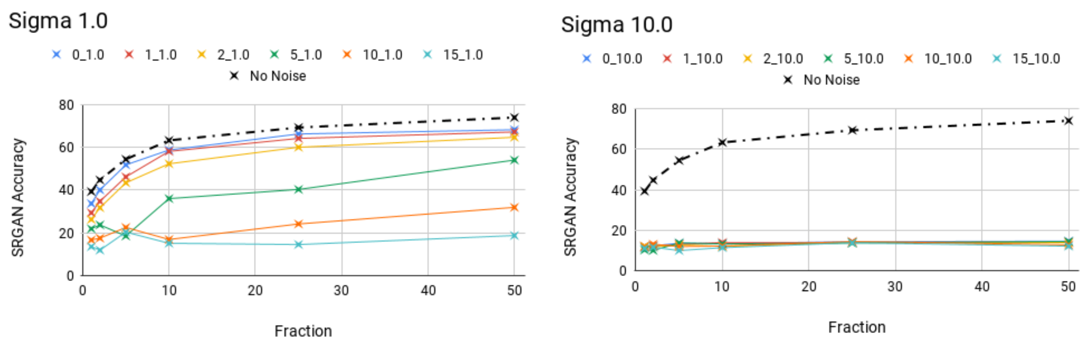


Figure 7.5: Performance of A2 for fixed σ and varying μ

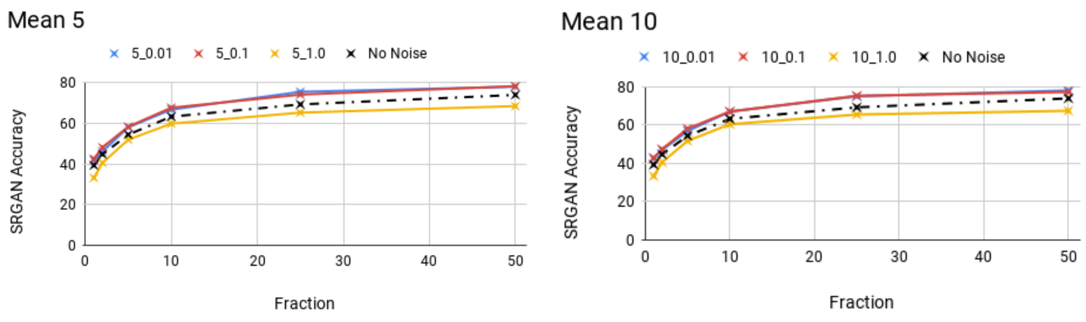


Figure 7.6: Performance of A3 for fixed σ and varying μ

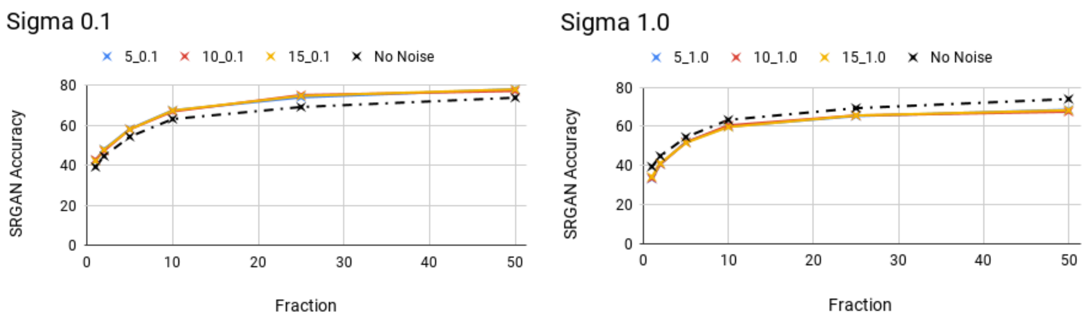


Figure 7.7: Performance of A3 for fixed σ and varying μ

7.3 Test (vs) SRGAN Accuracy for varying λ_1/λ_2

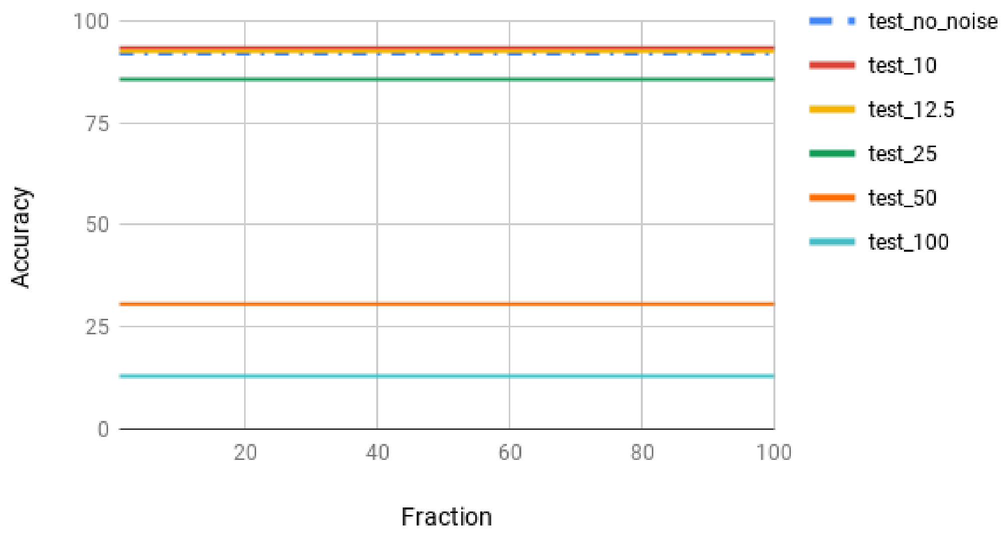


Figure 7.8: User performance for varying λ_1/λ_2

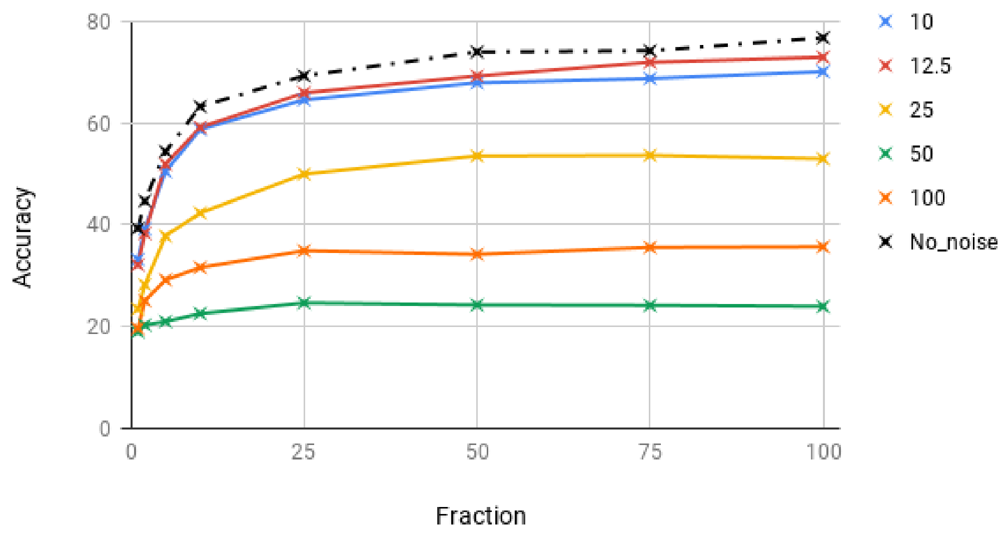


Figure 7.9: Adversary performance for varying λ_1/λ_2

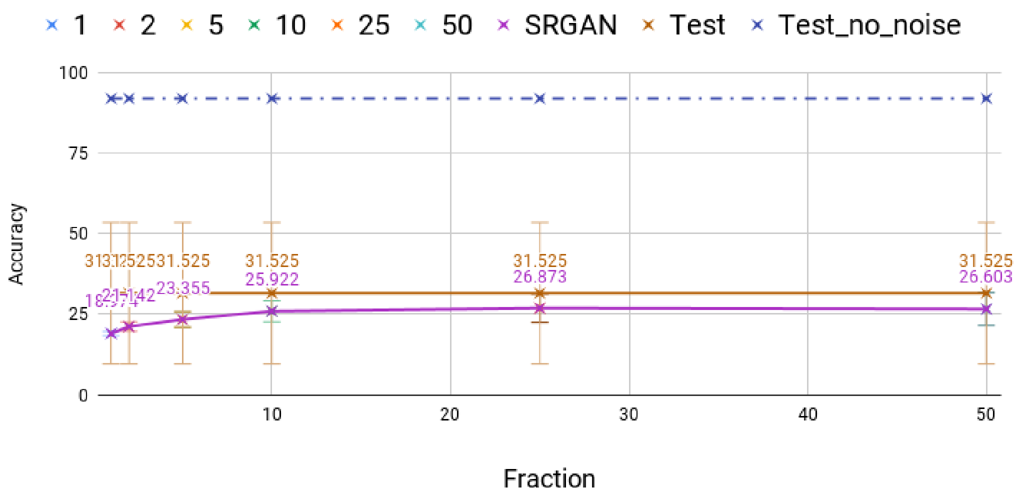


Figure 7.10: Comparing the performance of user and the adversary for λ_1/λ_2 = 50

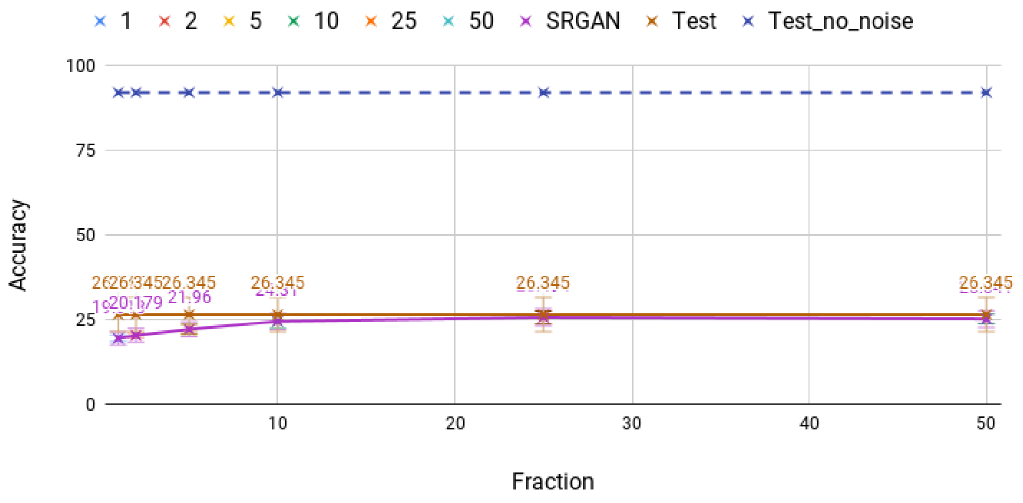


Figure 7.11: Comparing the performance of user and the adversary for λ_1/λ_2 = 100

CHAPTER 8

CONCLUSION

The growing interest in Deep Learning As a Service (DLaaS), where a marketplace of predictors is available on a pay-per-use basis, requires attention to the security and privacy of this model. Not all data-sets are sensitive, but in many applications in medicine, finance, and marketing the relevant data on which predictions are to be made is typically very sensitive.

We discussed the notion of privacy in the context of Deep Learning As a Service. We proposed the split framework and defined the goal of the adversary for the current task at hand.

We defined three different adversaries based on the information they have and their capabilities. We have shown the variation in their performance for one particular defensive mechanism. Finally we proposed a defensive mechanism against the most powerful adversary. Unlike previous works our framework is time and compute efficient as we are just including one extra layer which performs sampling and addition operations.

CHAPTER 9

FUTURE WORK

9.1 Shuffling

Recent experiments have shown that we get a significant decrease in the adversarial performance by just shuffling channels of the intermediate features, before sending them to M2. We have found that with increase in the number of channels at the split, performance of the adversary decreases. Work can be done on how the number of shuffling orders used affects the performance of adversary. This can further be discussed in a setting where we do not trust the server.

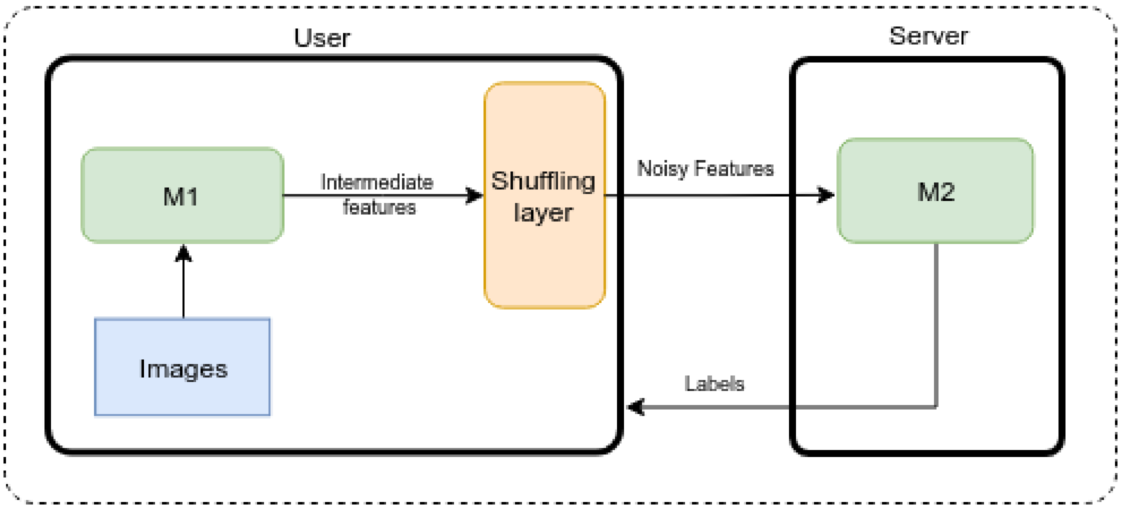


Figure 9.1: Shuffling framework

9.2 Keeping the labels also private

Instead of cutting the network at one single point if we cut the network at two different points such that a significant portion of the network in the middle is loaded onto the server whereas the initial and final parts are with the user itself. In this case we need not share the labels of our dataset with the server. This might further improve our privacy guarantees as the server does not the number of classes or the class labels of the user's dataset

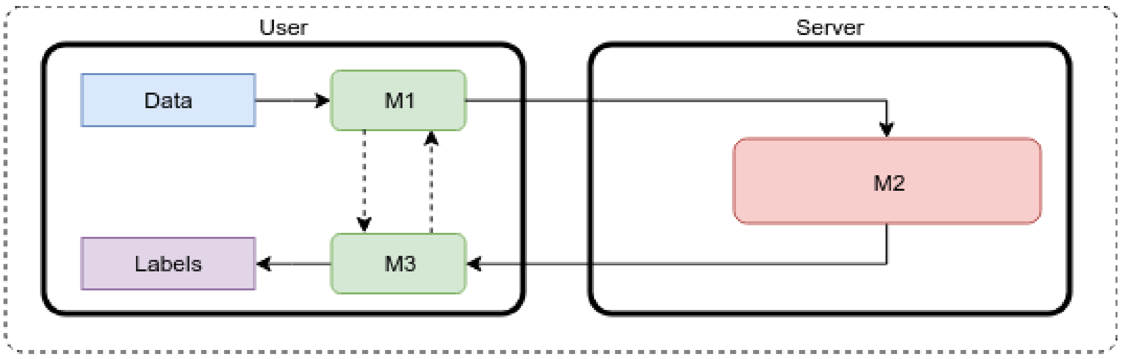


Figure 9.2: Double Split model

9.3 Effect of gradients on privacy

In the case where we deploy our framework during training, we can look into the fact that the adversary is also getting access to gradients sent from server to user. Will this harm user's privacy in any manner.

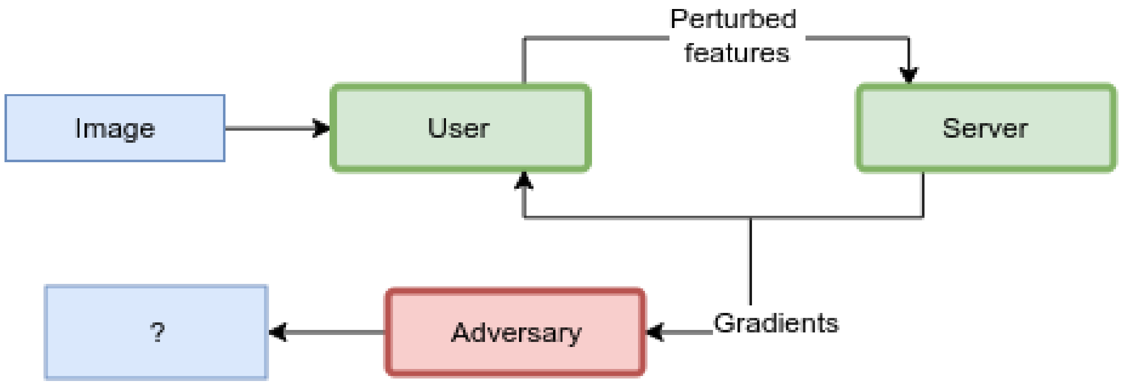


Figure 9.3: Attack on gradients

REFERENCES

[1] Agrawal, Rakesh and Srikant, Ramakrishnan. *Privacy-preserving data mining*. In *ACM Sigmod Record*, pp. 439–450.ACM, 2000

[2] Dwork, Cynthia. *Differential privacy*. In *Encyclopedia of Cryptography and Security*, pp. 338–340. Springer, 2011.

[3] Dowlin, Nathan and Gilad-Bachrach, Ran and Laine, Kim and Lauter, Kristin and Naehrig, Michael and Wernsing, John, *CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy*, Microsoft Research, 2016.

[4] E. Hesamifard, H. Takabi, and M. Ghasemi, *CryptoDL: Deep Neural Networks over Encrypted Data.* CoRR, vol. abs/1711.05189, 2017.

[5] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. 2017. *DeepSecure: Scalable Provably-Secure Deep Learning.* CoRR abs/1705.08963 (2017). *http://arxiv.org/abs/1705.08963*

[6] D. Bogdanov, S. Laur, and J. Willemson, *Sharemind: A Framework for Fast Privacy-Preserving Computations, in Computer Security - ESORICS 2008*

[7] M. S. Riazi, C. Weinert, O. Tkachenko, E. M. Songhori, T. Schneider, and F. Koushanfar. *Chameleon: A hybrid secure computation framework for machine learning applications. In ACM ASIACCS'18*. ACM Press, 2018.

[8] Payman Mohassel and Yupeng Zhang. *Secureml: A system for scalable privacypreserving machine learning*. IACR Cryptology ePrint Archive, 2017:396, 2017.

[9] Jian Liu, Mika Juuti, Yao Lu, and N Asokan. Oblivious neural network predictions via MiniONN transformations. *In ACM Conference on Computer and Communications Security (CCS)*, 2017

[10] Andrew Chi-Chih Yao. *How to generate and exchange secrets*. In Foundations of Computer Science, 1986., 27th Annual Symposium on, pages 162âĂŞ167. IEEE, 1986.

[11] Moni Naor and Benny Pinkas. *Computationally secure oblivious transfer. Journal of Cryptology*, 18(1):1–35, 2005.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep residual learning for image recognition*. In IEEE Conf. on Computer Vision and Pattern Recognition, pp. 770–778, 2016.

[13] Christian Ledig, Lucas Theis, Ferenc HuszÃąr, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. *Photo-realistic single image super-resolution using a generative adversarial network.* arXiv preprint arXiv:1609.04802, 2016.

[14] Jianfeng Dong, Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. *Unsupervised feature learning with symmetrically connected convolutional denoising autoencoders.* arXiv preprint arXiv:1611.09119, 2016.

[15] M. Li, L. Lai, N. Suda, V. Chandra, and D. Z. Pan, *Privynet: A flexible framework for privacy-preserving deep neural network training with a fine-grained privacy control.* arXiv:1709.06161, 2017.

[16] *https://github.com/kuangliu/pytorch-cifar*

[17] *https://github.com/aitorzip/PyTorch-SRGAN*

Appendix A

STRUCTURE OF NEURAL NETWORKS USED

A.1 VGG16

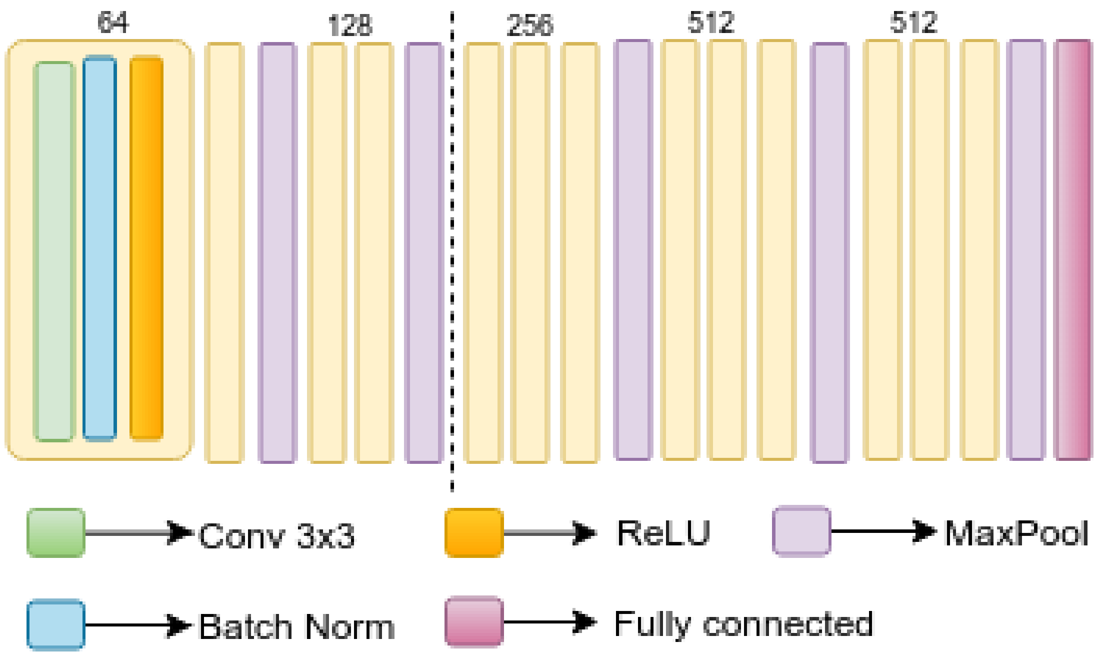


Figure A.1: VGG16 with the single split

A.2 Super Resolution Generative Adversarial Network (SRGAN)

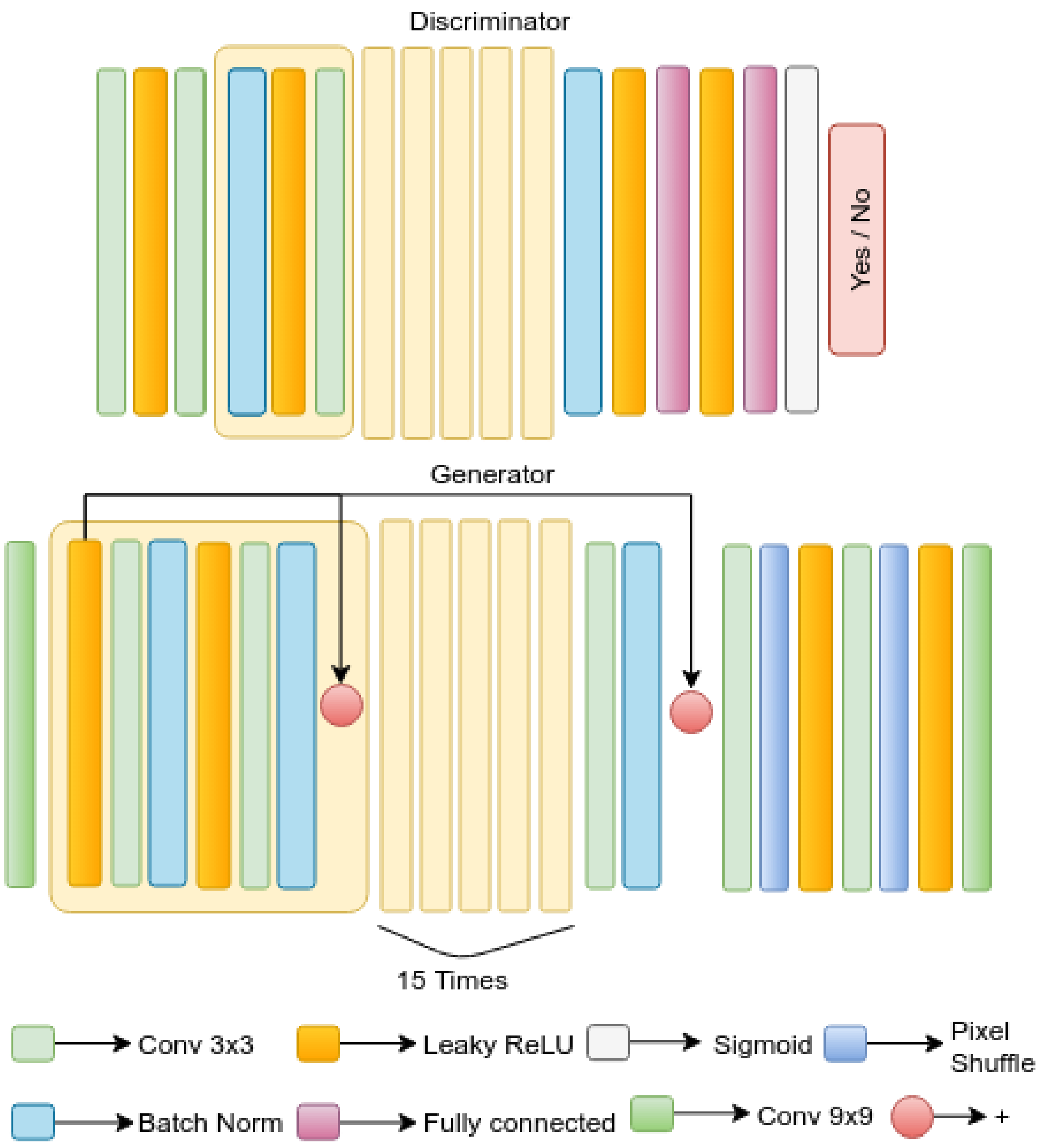


Figure A.2: Discriminator and Generator of SRGAN