Encryption-Robust CNN Model for Privacy-preserving Deep Learning

CAS771 Project Report

Tieyun Zhang, Shuiliang Wu

Department of Computing and Software, McMaster University, Hamilton, ON, Canada

E-mail: zhant124@mcmaster.ca; wus55@mcmaster.ca

Project GitHub: https://github.com/Shuiliangwu/CAS771-ERCNN

April 21, 2023

Table of Contents

1. Introduction

Privacy-preserving deep neural networks have made significant progress in recent years. The goal of privacy-preserving deep learning is to protect the privacy of data while still allowing it to be used for training and inference. This is especially important in fields like healthcare, finance, and retail, where personal information must be protected [1, 2].

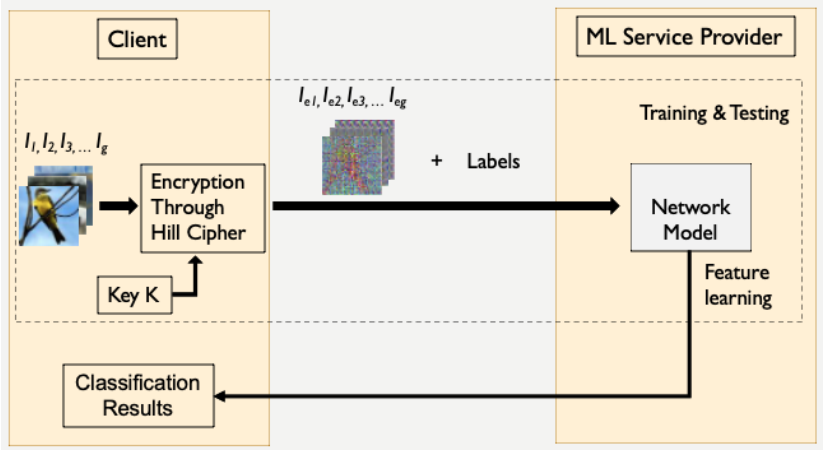

Figure 1-1. Project Scenario

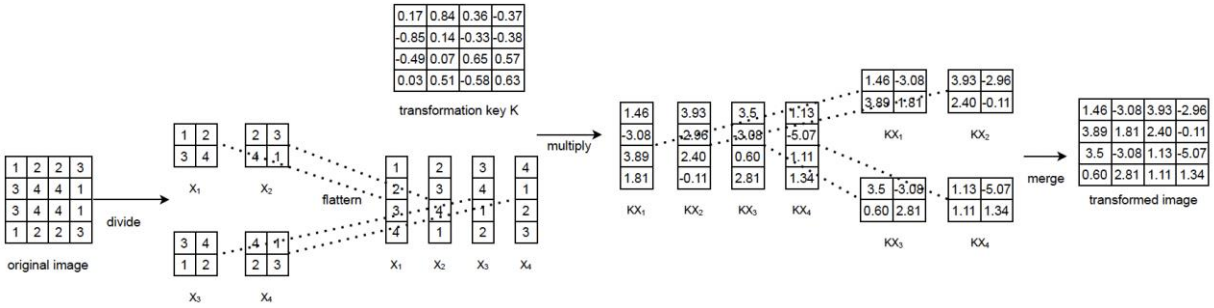

Figure 1-2 Image Encryption Through Linear Transformation

This project is basically mapping the real-world case of privacy-preserving deep learning. As shown in Figure 1-1, the client, who has limited capacity to classify the collected images, out-sources the job to a third-party machine learning service provider. To protect the image information, in this project client applied encryption on images through liner transformation which falls the category called Hill Cipher. Demonstration of encryption can be seen in Figure 1-2. Encryption is basically a split-merge process on the image blocks and linear transformation is performed on all the blocks through matrix multiplication. The output is an encrypted image of the same size. The

transformation key or cipher key was generated and kept at client side. Acting as the third-party machine learning service provider, our task is to develop an Encryption-Robust (ER) model to classify encrypted images sent from the client with a satisfied accuracy.

Table 1-1 Dataset Information

	Dataset 1	Dataset 2
Image Categories	10	50
Image Resolution	32 × 32	224 × 224
Block Size	4 × 4	8 × 8
Size of Transformation Key	3 × (16, 16)	3 × (64, 64)

The client provides two datasets for us to train and test the ER model. Each dataset contains training data, testing data and their label information. As summarized in **Table 1-1**, the provided two datasets are different in terms of image categories, image resolution, block size and size of transformation key. However, each image in datasets shares the same transformation key.

2. Related Work

Convolutional Neural Networks (CNNs) are widely used for image classification tasks and can be applied to encrypted images as well [1, 3]. However, this presents several challenges because traditional CNNs require the images to be decrypted before being processed.

The application of CNNs for the decryption of images is an area of research that aims to use neural networks to recover the original image from an encrypted representation.
One approach to achieve this is to train CNN to learn the inverse of the encryption process applied to the images. For example, if the images were encrypted using a classical encryption technique such as the Hill Cipher, the CNN can be trained to learn the inverse encryption process and decrypt the images [4].

Another approach is to use a generative model, such as a Generative Adversarial Network (GAN), to generate an image that is similar to the original image. The GAN can be trained on a dataset of

encrypted images and their corresponding original images to learn the mapping from the encrypted representation to the original image [5].

However, this area of research faces several challenges, such as ensuring that CNN can effectively learn the inverse encryption process or the mapping from the encrypted representation to the original image [4].

3. Proposed Method

3.1. The Convolutional Approach of Encryption

The hill cipher encryption can be described as a matrix multiplication of the original image (referred as "plaintext" in the following text) and the key matrix. However, it's also possible to describe the encryption process as a convolution of the plaintext and the key matrix.

The convolution of two matrices is defined as the sum of the element-wise multiplication of the two matrices. The convolution of two matrices is shown in the following equation:

$$C = A * B = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} B_{i,j}$$

where A and B are the two matrices to be convolved and C is the resulting matrix.

The convolution of the plaintext and the key matrix is similar. However, suppose the block size is k, we need to consider the fact that the key matrix is a square matrix of size $k^2 \times k^2$ and the blocks in the plaintext are square matrices of size $k \times k$. The hill cipher in this case reshapes the block in the plaintext into a vector of size $k^2 \times 1$ and multiplies it with each row of the key matrix. Therefore, we need to split the key matrix into k^2 submatrices of size $k \times k$ and convolve each submatrix with the plaintext. If we denote the submatrices of the key matrix as K_1, K_2, . . . , K_{n^2} then K_i is the vector of the ith row of the key matrix be reshaped into a $n \times n$ matrix. And we denote the ith block in the plaintext as Pi and the ith block in the ciphertext as Ci. Then the encryption process can be described as:

$$C_i = P_i * K_i$$

where i ranges from 1 to n^2.

3.2. The Convolution Approach of Decryption

The decryption process is the same to the encryption process except the key matrix for decryption is the inverse of the key matrix for encryption. Therefore, we can use the same convolutional approach to describe the decryption process.

The fact that decryption is also a convolution process means that we can use the convolutional neural network to decrypt the ciphertext. The convolutional neural network is a type of neural network that is used to process images. If we arrange the convolutional layer carefully, we can use the convolutional neural network to decrypt the ciphertext. Our proposed method is to add an adaptation network before the normal convolutional neural network and the classification network. The adaptation network is used to learn the key matrix and recover more learnable features from the ciphertext. The normal convolutional neural network is used to recover the plaintext from the ciphertext. The classification network is used to classify the plaintext. The architecture of the proposed method is shown in Figure 3-1.

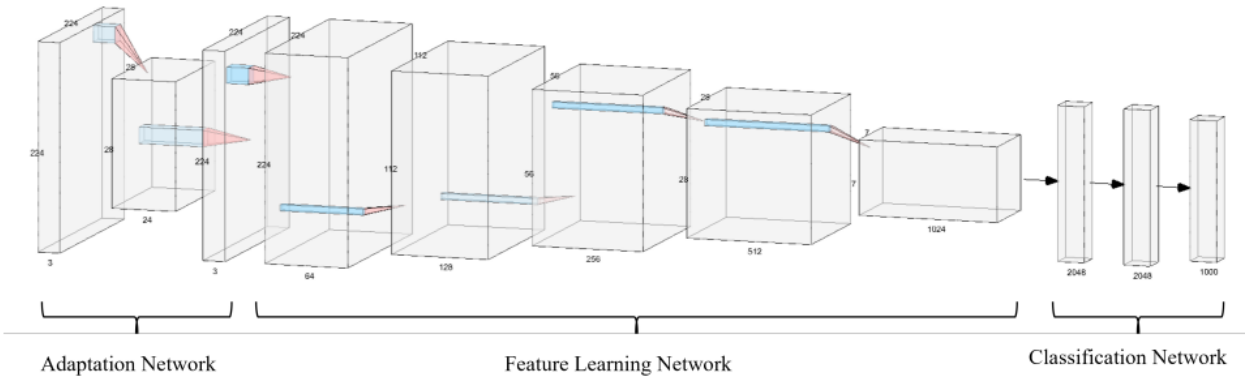

Figure 3-1 The architecture of the proposed method.

3.3. The Adaption Network

The adaptation network is used to learn the key matrix and recover more learnable features from the ciphertext. The adaptation network is a convolutional neural network that is used to learn the key matrix. The architecture of the adaptation network is shown in Figure 3-2.

The first layer of the adaptation network is a convolutional layer. It's used to perform the convolution of the ciphertext and the inversed key matrix, where the inversed key matrix is to be learned during the training process. The case shown in Figure 3-2, the block size k is 8, thus we choose the kernel size of the convolutional layer to be 8. And since the convolution for encryption/decryption doesn't overlap, we choose the stride of the convolutional layer to be 8. We choose the number of filters to be 192 since the original image is 224 × 224 with 3 channels and the block size is 8, thus the key matrix has 224/8 × 224/8 × 3 = 192 different rows, which is the number of filters we choose for the convolutional layer.

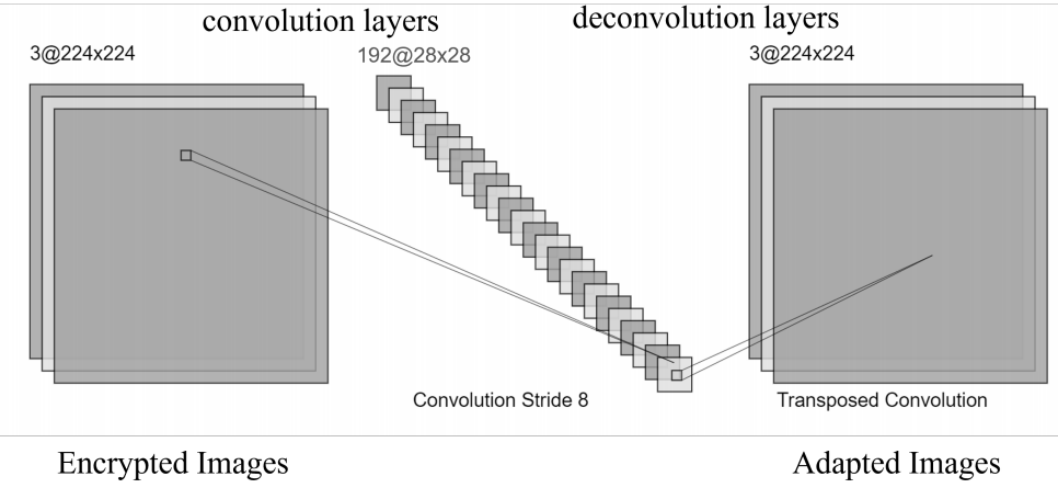

Figure 3-2 The architecture of the adaptation network.

The second layer of the adaptation network is a deconvolutional layer. It's used to resize the output of the convolutional layer with deep feature maps to the original size of the input image. Most of the pretrained CNN models require an input of a 3-channel image. And CNN usually extracts the spatial information from the input image. Therefore, we choose to recover the spatial information by upsampling the output of the convolutional layer. The upsampling is done by the deconvolutional layer. Since the convolutional layer produces 192 feature maps, the deconvolutional layer will learn a correct combination of different features to produce more pixels. In theory, total pixels of the output of the convolutional layer is the same of the original image, thus the deconvolutional layer will learn to recover the original image from the output of the convolutional layer.

3.3 Pre-trained CNN Model Selection

ResNet, DenseNet, ResNeXt, and ConvNeXt are all deep neural network architectures that have been developed to improve the performance of CNNs in image classification tasks. Their pros and cons are summarized in **Table 3-1** below. In this project, eight pre-trained models were applied on this task, which includes ResNet-34, ResNet-50, DenseNet, ResNeXt-50, ConvNeXt-Tiny, ConvNeXt-Small, ConvNeXt-Base and ConvNeXt-Large.

Table 3-1 Summary of selected CNN models

Architecture	**Year Introduced**	**Pros**	**Cons**
ResNet Family	2015 by Microsoft	- Address the problem of vanishing gradients - Can enable deeper network architectures	- May have more parameters to learn than other architectures - May be sensitive to initialization
DenseNet	2016 by Tsinghua University and Microsoft	- Efficient feature reuse - Can learn more complex representations	- High memory usage - Can be computationally expensive - May be sensitive to batch size
ResNeXt50	2016 by Facebook	- Can learn more diverse features	- May require more computational resources - Limited number of cardinalities
ConvNeXt Family	2020 by Carnegie Mellon University	- Combines convolutional and fully connected layers - Adaptively combines different features from different layers to improve performance	- May require the most computational resources

4. Results

4.1 Hyperparameter Tuning

4.1.1 Batch Size

Batch size was optimized through performance comparison on dataset1 using ResNet-34 and ResNet-50. As shown in Figure 4-1, in the range of 16 to 256 on batch size, the best validation accuracies achieved by ResNet-50 are higher than those achieved by ResNet-34. The peak is achieved at batch size of 32 on ResNet-50 meanwhile for ResNet-34, the accuracies decrease with the increase of batch size. Considering performance and computational cost (computation time is shorter when using a higher batch size), batch size was optimized as 32. The optimization was further verified by RexNeXt-50 and ConvNeXt-Base using different learning rates, which can be seen in Figure 4-2 and 4-3. It is clearly demonstrated that batch size of 32 provides the best performance for both RexNeXt-50 and ConvNeXt-Base. Therefore, batch size is fixed at 32 for the rest of the project.

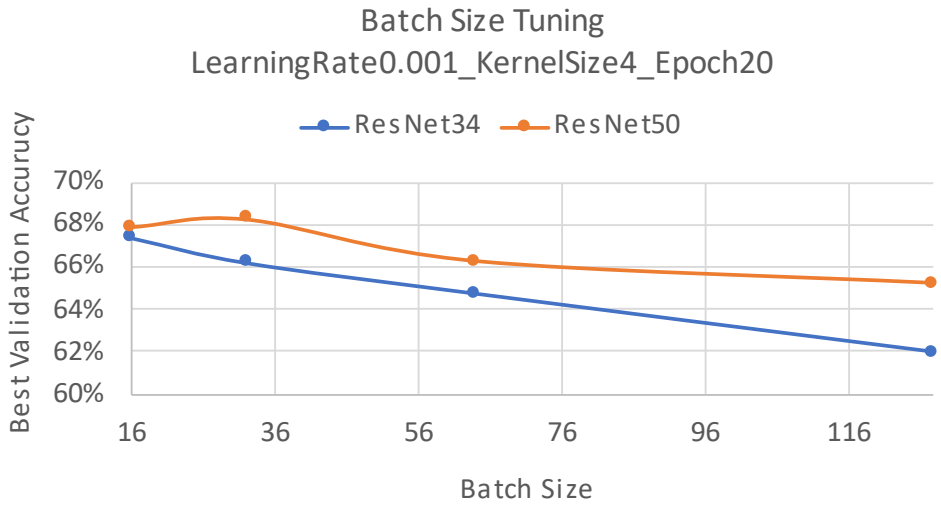

Figure 4-1 Batch Size Tuning Through Dataset1 Using ResNet

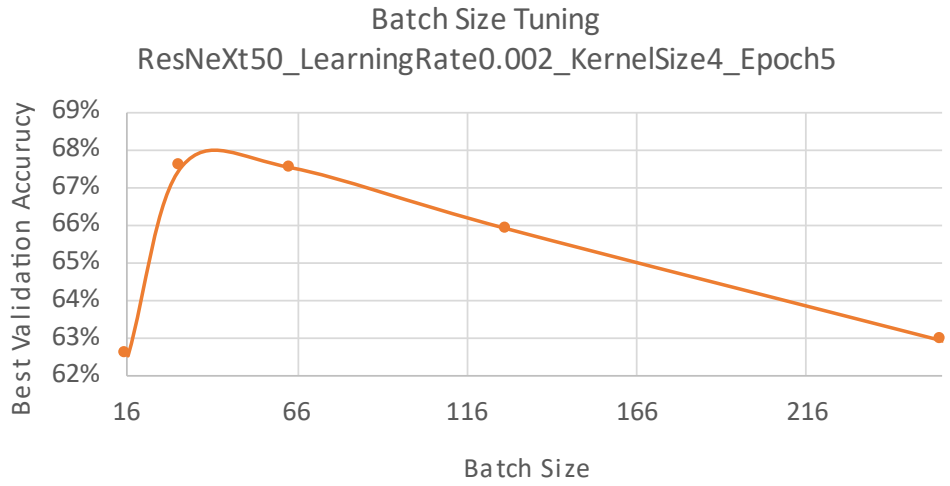

Figure 4-2 Batch Size Tuning Through Dataset1 Using ResNeXt-50

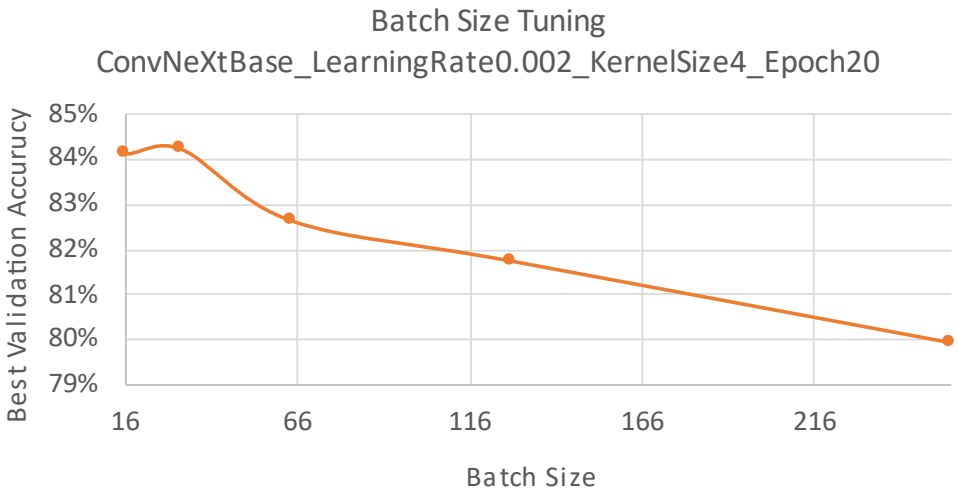

Figure 4-3 Batch Size Tuning Through Dataset1 Using ConvNeXt-Base

4.1.2 Learning Rate

Learning rate was optimized through dataset1 using ResNet-50. As shown in Figure 4-4, the learning rate of 0.002 outperforms the rest. Thus, for the rest of the project, the learning rate is fixed at 0.002.

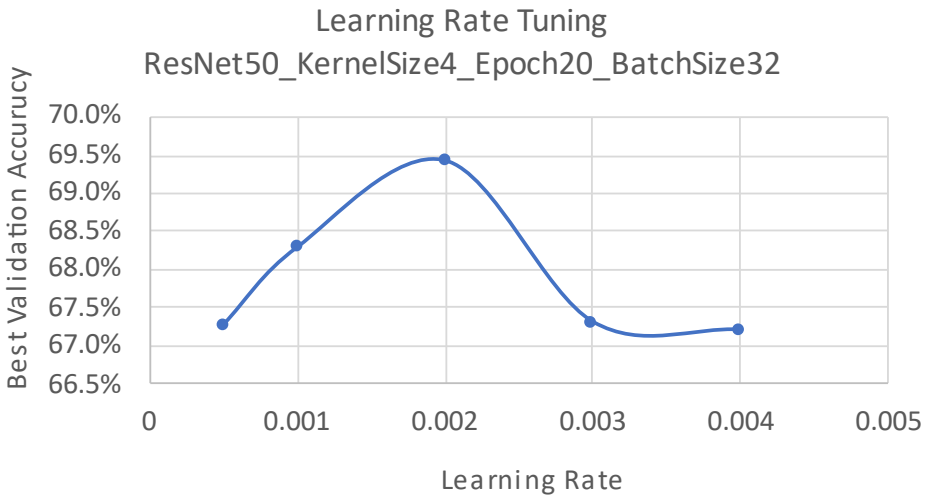

Figure 4-4 Learning Rate Tuning Through Dataset1 Using ResNet-50

4.1.3 Kernel Size

As shown in Figure 4-5, compared with kernel size of 2 and 8, kernel size of 4 achieves the best performance on dataset1 using ResNet-50. The kernel size is fixed at 4 for dataset1 and 8 for dataset2 for the rest of the project.

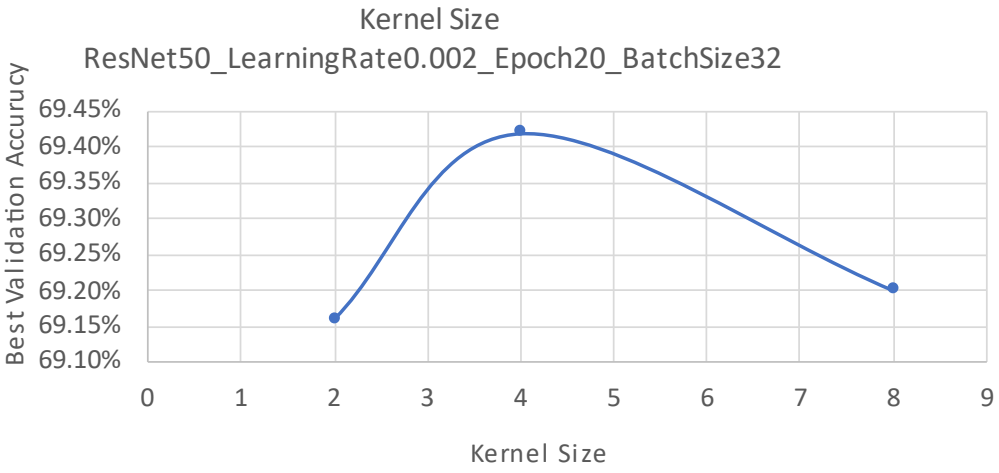

Figure 4-5 Kernel Size Tuning Through Dataset1 Using ResNet-50

4.2 Model Comparison

The model comparison results on dataset1 are summarized in Figure 4-6. It is indicated that ConvNeXt family outperforms ResNet-34, ResNet-50, DenseNet and ResNeXt-50. All the pre-

trained models perform better than their modified versions with additional adaptation layers, which indicates that adaptation layers play a negative role of feature learning on dataset1. The best validation accuracy on dataset1 is 84.25% achieved by ConvNeXt-Base of 20 Epochs.

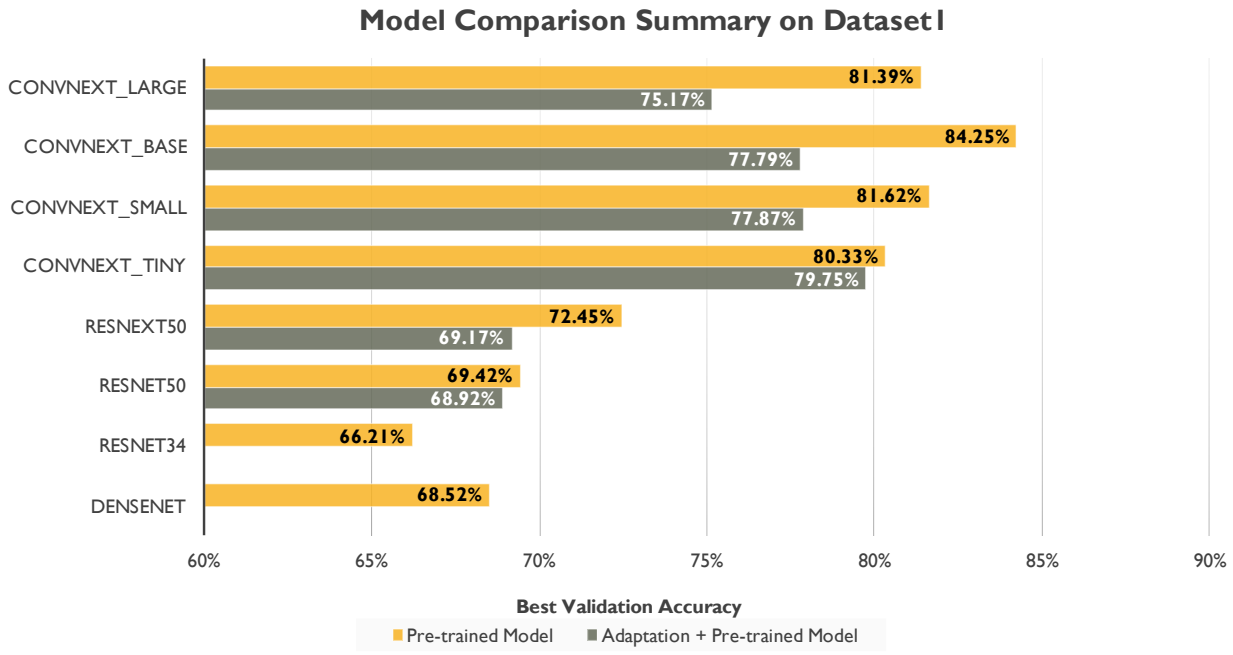

Figure 4-6 Model Comparison Summary on Dataset1

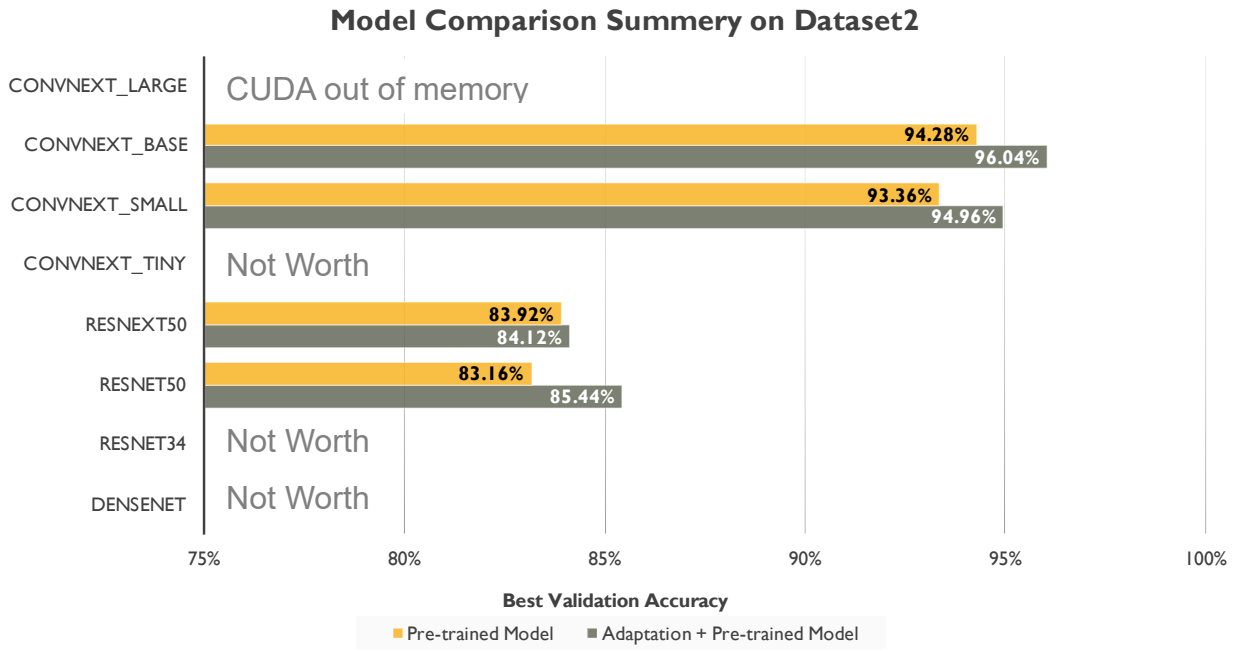

Figure 4-7 Model Comparison Summary on Dataset2

The model comparison results on dataset2 are summarized in Figure 4-7. Same as what was found on dataset1, ConvNeXt family outperforms ResNet-50 and ResNeXt-50. ConvNeXt-Large cannot be applied on dataset2 due to the limitation of GPU memory (10GB, RTX3080). ConvNeXt-Tiny, ResNet-34 and DenseNet were not performed as they highly possibly won't contribute to the best validation accuracy. Results from the performed model indicate that additional adaptation layers provide positive effects on feature learning on dataset2. The best validation accuracy on dataset2 is 96.04% achieved by ConvNeXt-base of 20 Epochs. As shown in Figure 4-8, with adaptation layer, training accuracy and validation accuracy are around 90% which is significantly higher than those of pre-trained ConvNeXt without adaptation layer.

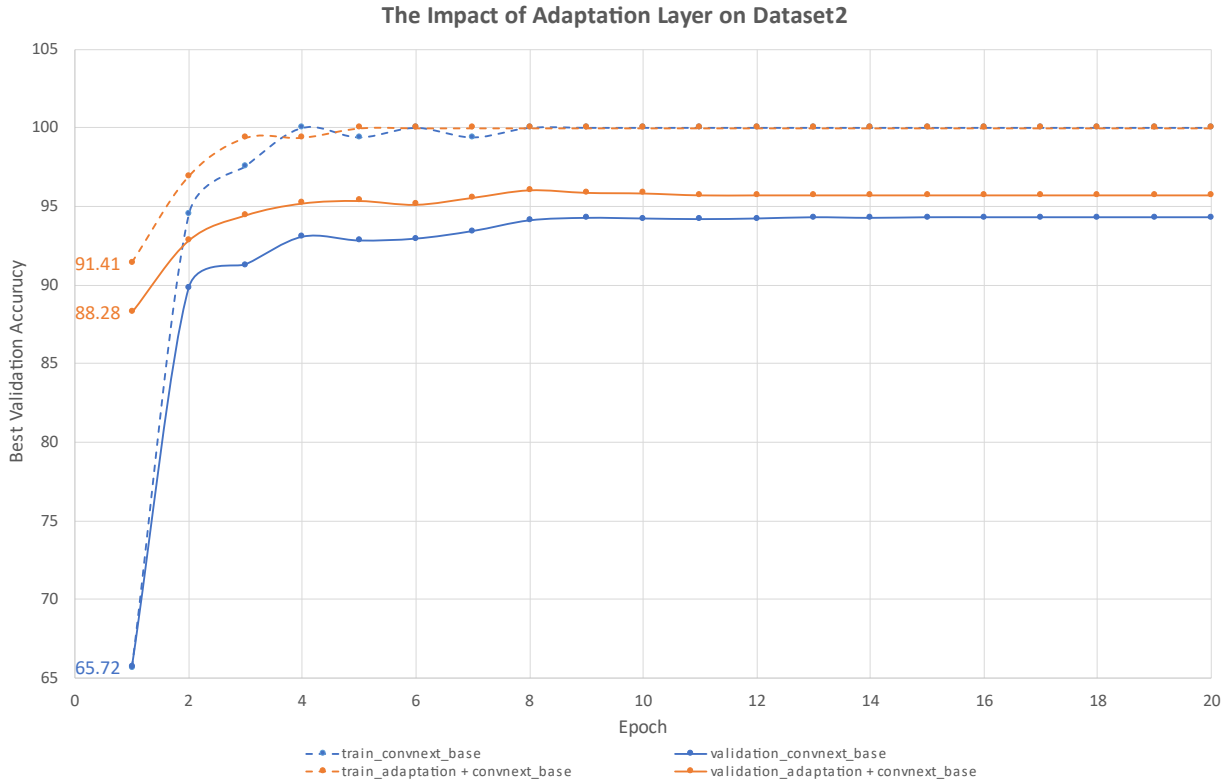

Figure 4-8 The Impact of Adaptation Layer on Dataset2

5. Decryption

The original design of our model is supposed to have decryption ability. We have engineered the adaptation layers according to the encryption process and expected that it would recover part of the encrypted information as shown in Figure 5-1.

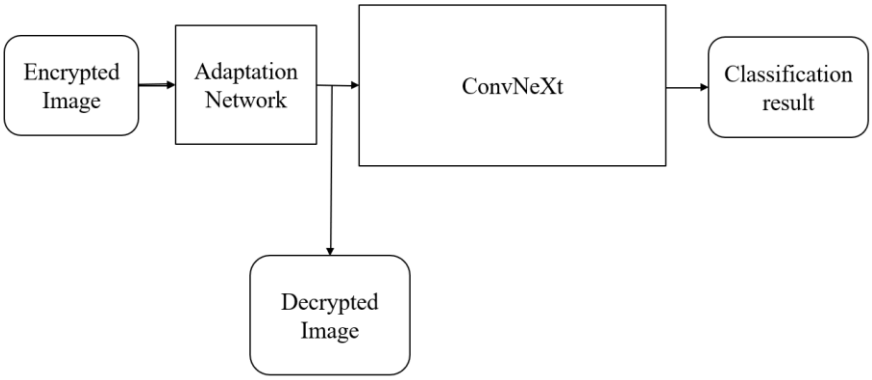

Figure 5-1 Expected decryption framework of our model

Our experiment of image decryption is basically using a trained adaptation network as the decryptor, and simply showing output image produced by the adaptation network.

5.1 Initial Result of Decryption

Without knowing the encrypted image is encrypted from ImageNet, which the ConvNeXt model is pretrained on, the initial result of training is unexpectedly good. As shown in Figure 5-2, after several epochs of training, we can easily distinguish objects from the output images of the adaptation network.

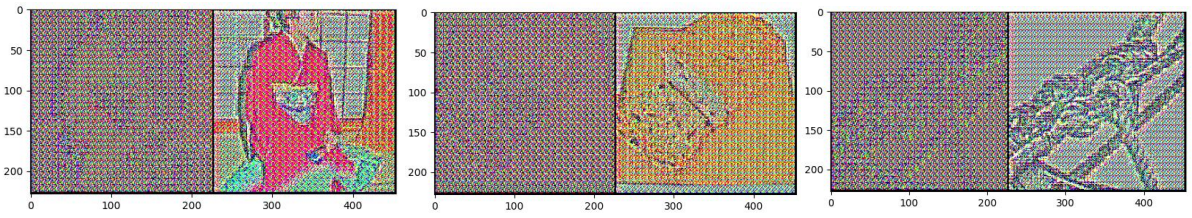

Figure 5-2 Samples of images outputted by adaptation network. The left image of each subfigure is the encrypted image, the one on the right is the decrypted image.

From reading the training curves, we observed that the model converged very quickly, leaving a short period of time for the adaptation to get well trained. The possible underlying reason is that the learning ability of ConvNeXt is too strong compared to our adaptation network, it also undertook some of the decryption process in the end-to-end training. To cope with this issue, we came up with a multi-round training pipeline as shown in Figure 5-3.

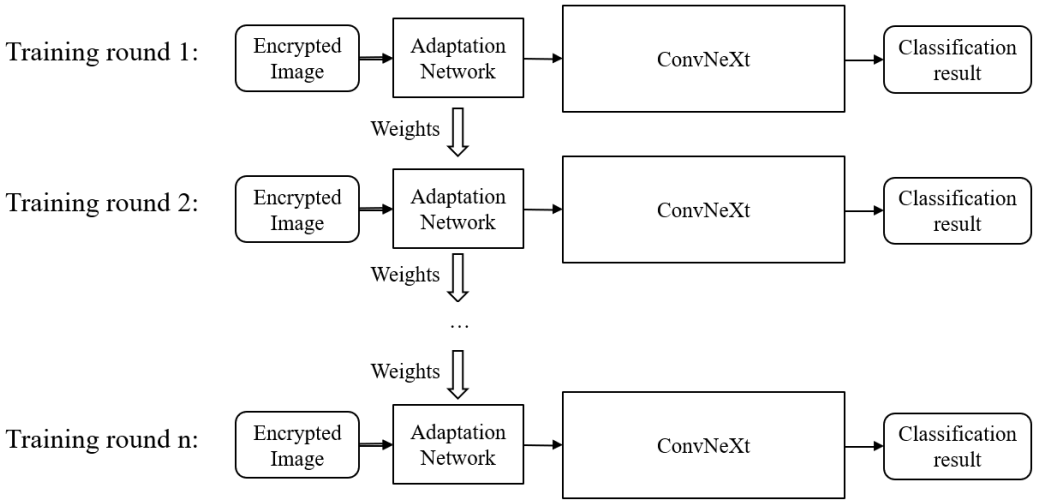

Figure 5-3 The pipeline of the multi-round training scheme

During each round we perform a regular 5 epoch training. At the end of each round, we save the model and end the training. At the beginning of each round, we only load the adaptation network from the model saved in the last round and load a fresh ConvNeXt model. By this approach we can give the adaptation layer more time to get fine turned and well trained. We saw a significant improvement in the quality of the decrypted image as shown in Figure 5-4. The approach of multiple rounds of training is proved to be effective.

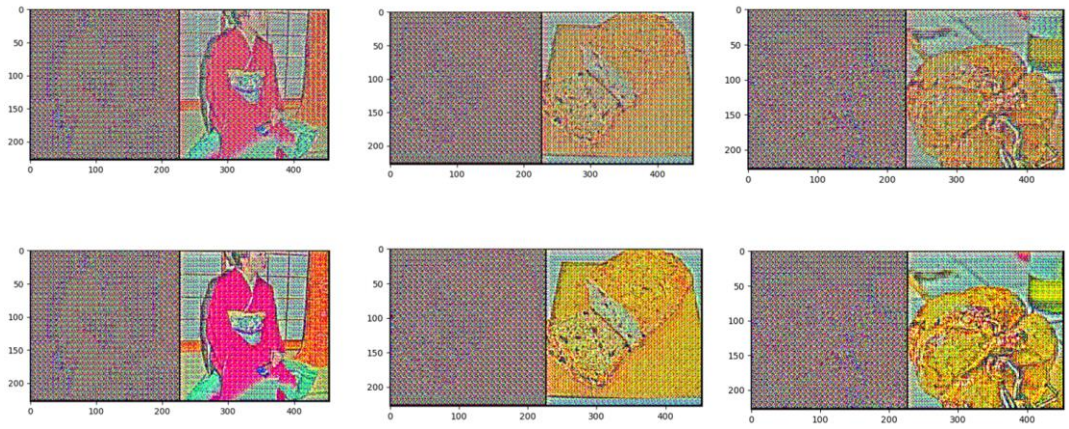

Figure 5-4 Comparison between the results of single round training and multi-round training. The upper three images are the result of the single round training.

However, this experiment has a flaw that the pretrained CNN model has already been trained on the images that were encrypted. It's possible that the CNN model memorized all of the original images and forced the adaptation network to learn the encryption key.

5.2 Solving the Pretraining Issue

After acknowledging that using an ImageNet pretrained model could introduce flaws in our experiment, we have tried two different approaches to solve this problem. We still used the pretrained CNN model, however, we made sure that the model is not pretrained on the images that were used to encrypt. The reason is that a pretrained CNN model has the ability to extract features from the unencrypted image. As we trained our model on an end-to-end basis, the ability of CNN to process plain image will force the adaptation network to decrypt the image.

5.2.1 Approach One: Using Self-pretrained ResNet50 Model

Our first attempt is to pretrain the model by ourselves. ImageNet1k dataset has 1000 classes, each class has more than 1000 images. It provides us with abundant images. Thus, we pretrained a ResNet50 model using the first 600 classes from ImageNet1k. Then we generated the encrypted image dataset **using an encryption key of size 16** from the 601 to 700 classes. After that we loaded the pretrained ResNet50 model and trained the model on the encrypted image. The approach can assure that the pretrained CNN model does not have any prior knowledge of

the original encrypted images since the pretraining dataset and encryption images are from different classes as shown in the Venn diagram in Figure 5-5.

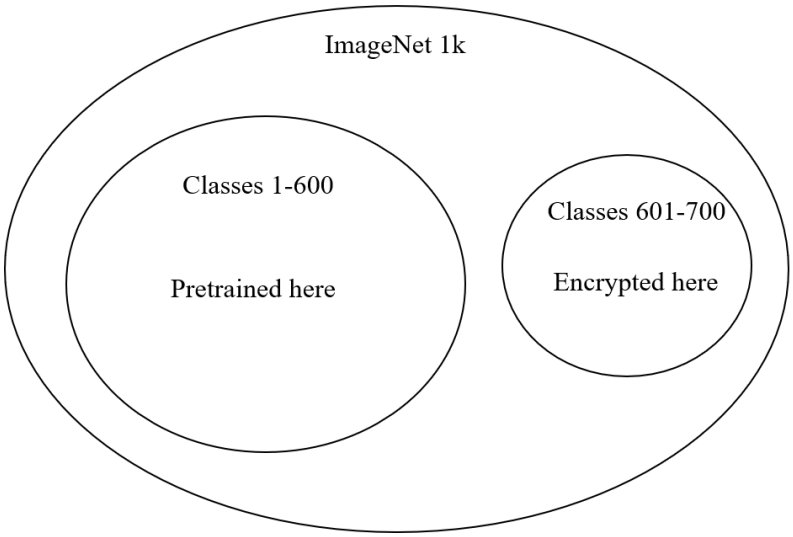

Figure 5-5 The Venn diagram showing the relationship between the pretraining dataset and encryption dataset

5.2.2 Approach Two: Using Images from ImageNet21k

Our second attempt is to resort to the larger dataset ImageNet21k, which is a superset of ImageNet1k. ImageNet21k is allowed to be partially downloaded. Each class in ImageNet has a unique "synset" number, which is an identifier used by both ImageNet1k and ImageNet21k. By leveraging the metadata of two datasets, it is easy for us to sift the classes in ImageNet21k that aren't in ImageNet1k. Thus, we can safely use the CNN model pretrained on ImageNet1k and do the decryption on the classes that are not in ImageNet1k. The relationship between the two datasets is shown in the Venn diagram in Figure 5-6.

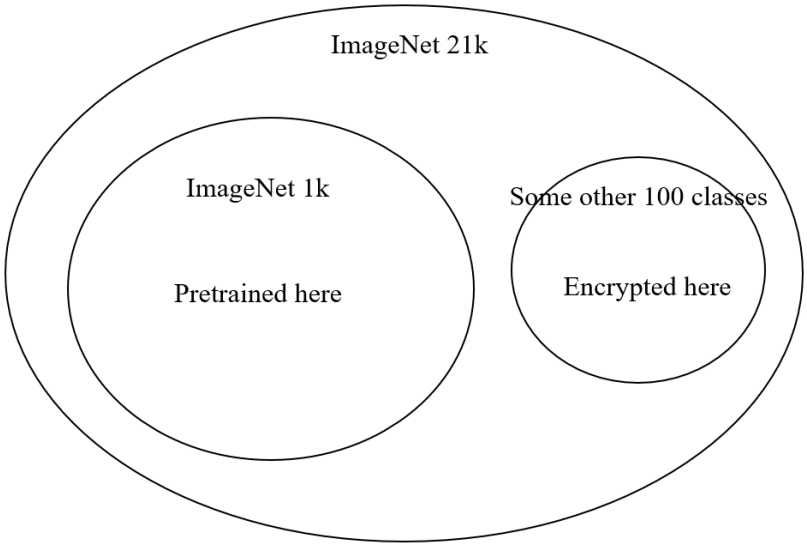

Figure 5-6 The Venn diagram showing the relationship between the pretraining dataset ImageNet 1k and encryption dataset

This approach is better because pretraining CNN on ImageNet is difficult. It needs a large cluster of GPUs and good tuning expertise. By using the well pretrained CNN model and generating the encrypted image on different classes we can hopefully get a better result.

5.3 Results

5.3.1 Results of Approach One

In the first approach which is to pretrain ResNet50 by us, the best validation accuracy we achieved is 57.46%. This result is not optimized as no tuning was applied. In order to finish the pretraining on 600 classes within a reasonable amount of time, the only set of hyperparameters used is 128 batch size, 0.004 learning rate, 16 key size and 20 epochs. The best pretraining model was saved and later used together adaptation layer to do decryption. A multi-round training strategy was applied (pipeline is shown in Figure 5-3) on training of encrypted images and its accuracy results can be seen in below Figure 5-7. The best validation accuracy we achieved through multi-round training is 56.60% which was saved as the best model for decryption. As shown in Figure 5-7, although multi-round training strategy seems doesn't enhance accuracy significantly, it shall provide adaptation layer enough opportunities to be trained on image decryption.

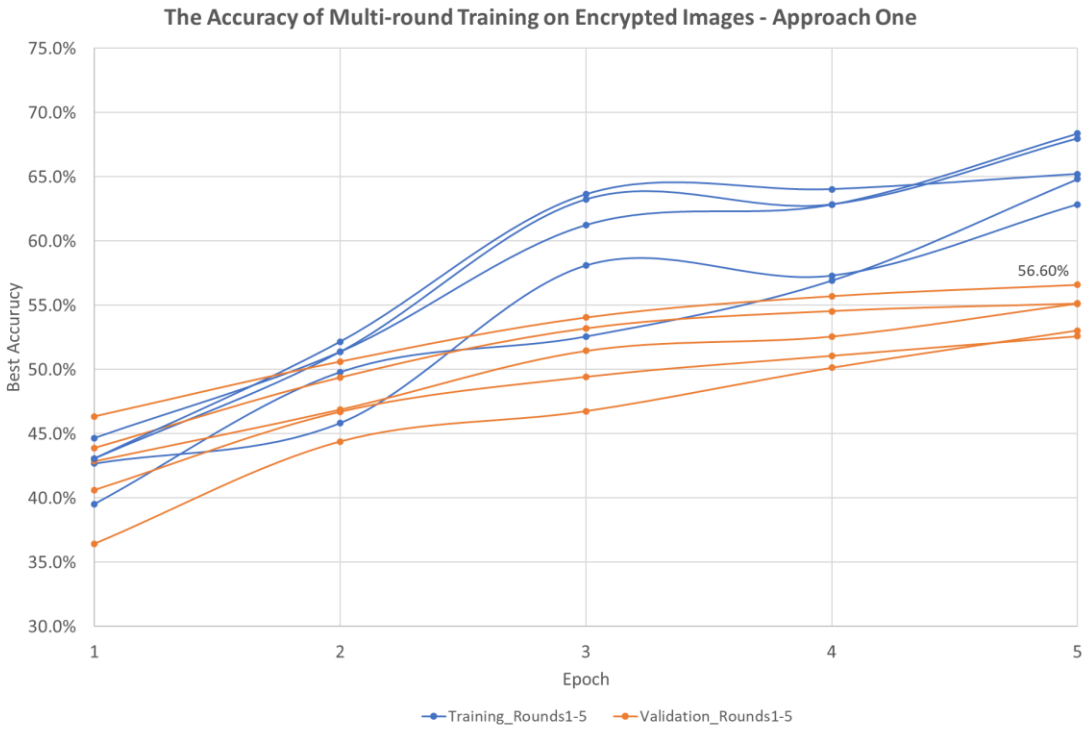

Figure 5-7 Accuracy results of decryption using Approach One

In Figure 5-8, The randomly selected images were shown below as the decryption results. The decrypted images don't provide details as good as our initial result of decryption (shown in Figure 5-4), however, considering our model is not pretrained very well and the encryption key is larger (size = 16), images are still quite identifiable. It provides evidence that after multi-round training, the adaptation layer can indeed play an effective role in image decryption.

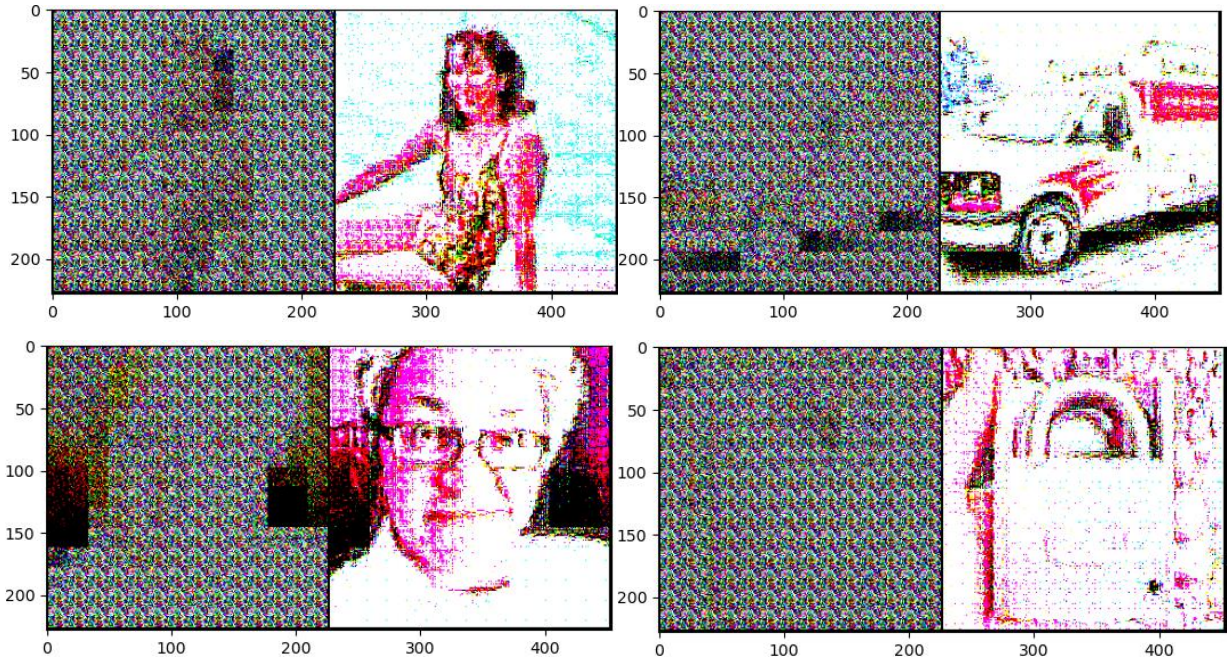

Figure 5-8 Random selected decryption results from Approach One (left: encrypted image; right: decrypted image)

5.3.2 Results of Approach Two

Only two rounds of five epochs each were performed on the second approach due to the limitation of time and calculation power. The previous best model on dataset 2, ConvNeXt-Base, was used in this approach. As shown in Figure 5-9, it is clearly overfitting and the best accuracy achieved is 52.31%. However, it doesn't provide good decryption in this approach.

One possible reason we observed is that there are multiple kinds of objects within a single class (like dogs, cats, foxes are in a single class) in the datasets from ImageNet21k, posing a challenge for our model to distinguish between different classes. Because accurate label information plays a crucial role, as it's the only information that could be used to train the adaptation network.

Another possibility is that our training time is too limited. As training ConvNeXt is extremely time consuming, we didn't run enough epochs and rounds to get the best results. In theory this approach could be more promising. A possible solution would be using ResNet50 as the pretrained CNN as it's considerably easier to train. It's also useful to further sift the ImageNet 21k classes based on

it's label quality. As there are plenty of classes we can use, we can choose those classes containing a single type of object within a class.

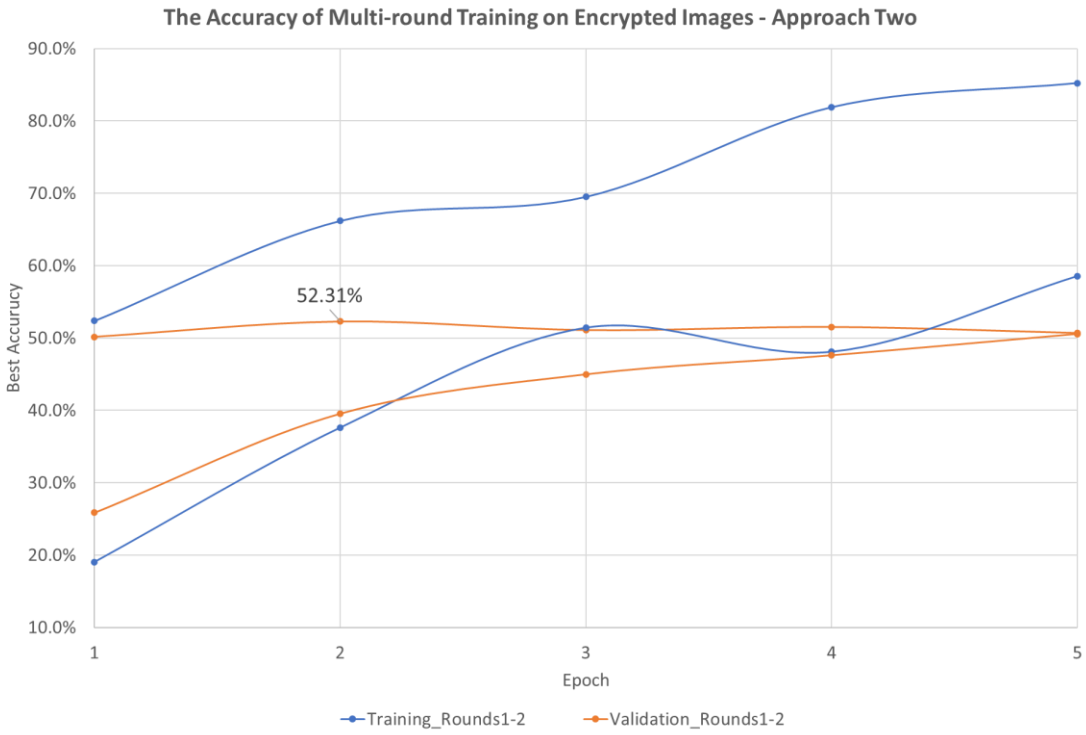

Figure 5-9 Accuracy results of decryption using Approach Two

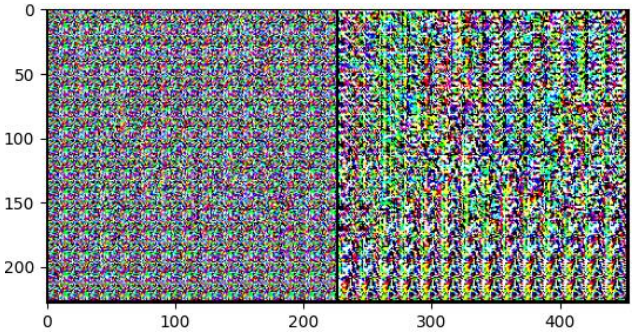

Figure 5-10 Failed decryption results from Approach Two (left: encrypted image; right: decrypted image)

5.4 Conclusion

In this project, we performed both roles as a service provider and an eavesdropper. We proposed an Encryption-Robust (ER) model for privacy-preserving deep learning. The ER model is based on a regular CNN classification model and an adaptation network. The adaptation network was

developed using convolution to perform the decryption. In total eight pre-trained models applied in this project, ConvNeXt-Base outperformed others and achieved highest accuracy values for both dataset 1 (84.25% as the highest for dataset 1) and dataset 2 (94.28%) provided. Adaptation network and multi-round training strategy were then applied with ConvNeXt-Base to do the decryption. It achieved an improvement on decryption on dataset1 and performed outstandingly on dataset 2. Moreover, the accuracy value was further increased to 96.72% (as the highest for dataset 2) with the application of adaptation network and multi-round training strategy.

A flaw was realized that pretrained ConvNeXt-Base has already seen and memorized all of the original images from dataset 2 thus the results on dataset 2 were possibly not solid. To ensure no-overlapping on images for pretraining and decryption, two different approaches were developed. The first approach using self-pretrained ResNet50 model showed that adaptation network after multi-round training can have an effective decryption capacity. The second approach using images from imageNet21k doesn't show good decryption. One possible reason is that the dataset used in approach two is more complicated with multiple types of objects contained in a single class. Time limitation is another reason.

6. References

[1] El Saj, Raghida, et al. "Privacy-preserving deep neural network methods: computational and perceptual methods—an overview." Electronics 10.11 (2021): 1367.

[2] Sirichotedumrong, Warit, et al. "Privacy-preserving deep neural networks with pixel-based image encryption considering data augmentation in the encrypted domain." 2019 IEEE International Conference on Image Processing (ICIP). IEEE, 2019.

[3] Momeny, Mohammad, et al. "A noise robust convolutional neural network for image classification." Results in Engineering 10 (2021): 100225.

[4] Lidkea, Viktor M., Radu Muresan, and Arafat Al-Dweik. "Convolutional neural network framework for encrypted image classification in cloud-based ITS." IEEE Open Journal of Intelligent Transportation Systems 1 (2020): 35-50.

[5] Charan, Deanne. "Generative Adversarial Networks for Classic Cryptanalysis." (2021).