DEFENSIVE QUANTIZATION LAYER FOR CONVOLUTIONAL NETWORKS AGAINST ADVERSARIAL ATTACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent research has intensively revealed the vulnerability of deep neural networks, especially for convolutional neural networks (CNNs) on the task of image recognition, through creating adversarial samples which "slightly" differ from legitimate samples. This vulnerability indicates that these powerful models are sensitive to specific perturbations and cannot filter out these adversarial perturbations. In this work, we propose a quantization-based method which enables a CNN to filter out adversarial perturbations effectively. Notably, different from prior work on input quantization, we apply the quantization in the intermediate layers of a CNN. Our approach is naturally aligned with the clustering of the coarse-grained semantic information learned by a CNN. Furthermore, to compensate for the loss of information which is inevitably caused by the quantization, we propose the multi-head quantization, where we project data points to different sub-spaces and perform quantization within each sub-space. We enclose our design in a quantization layer named as the Q-Layer. The results obtained on MNIST and Fashion-MNSIT datasets demonstrate that only adding one Q-Layer into a CNN could significantly improve its robustness against both white-box and black-box attacks.

1 INTRODUCTION

In recent years, along with the massive success of deep neural networks (DNNs) witnessed in many research fields, we have also observed their impressive failures when confronted with adversarial examples, especially for image recognition tasks. Prior work (Szegedy et al. (2014); Goodfellow et al. (2015)) has demonstrated that an adversarial image can be easily synthesized by adding to a legitimate image a specifically crafted perturbation, which is typically imperceptible for human visual inspection. The generated adversarial image, however, is strikingly effective for causing convolutional neural network (CNN) classifiers to make extreme confident misclassification results. This vulnerability of DNNs has stimulated the unceasing arms race between research on both attacking (Goodfellow et al. (2015); Kurakin et al. (2017); Carlini & Wagner (2017); Moosavi-Dezfooli et al. (2016); Chen et al. (2017); Brendel et al. (2018)) and defending (Madry et al. (2018); Samangouei et al. (2018b); Buckman et al. (2018); Zhang & Liang (2019)) these powerful models.

Among much existing work and a large variety of defense methods, several prior studies (Xu et al. (2018); Buckman et al. (2018); Zhang & Liang (2019)) have spent concerted efforts on defending adversarial attacks through *input quantization*. The principle idea of these methods is to use quantization to filter out small-scale adversarial perturbations. Recall that in prior work (Bau et al. (2017); Zeiler & Fergus (2014); Zhou et al. (2015)), it has been shown that the shallow layers of a CNN mostly capture fine-grained features including lines and curves. In the meantime, deeper layers learn coarse-grained yet semantically more critical features, which essentially discriminate different samples. Especially for classification tasks, it is natural to expect samples with the same classification label to share similar semantic information. As such, the semantic similarity between samples may be better revealed if we attend to their latent features learned by the intermediate layers of a CNN. Here we hypothesize that data points with similar semantic information should be distributed densely in the latent feature space. Thus, in order to more effectively filter out adversarial perturbations, we propose an alternative approach which quantizes the data representations embedded in the feature space produced by the intermediate layers of CNN classifiers. Interestingly, there have been other studies that develop similar approaches but for different purposes. For example, Wang et al. (2017; 2015) have applied k-means clustering on the intermediate feature maps of CNN models

to discover explainable visual concepts. Recent methods, including VQ-VAE (van den Oord et al. (2017)) and SOM-VAE (Fortuin et al. (2019)), were proposed to construct generative models for images and time-series data with discrete latent representations, which offer better explainability. However, to the best of our knowledge, the approach of applying intermediate layer quantization for CNN models has not been explored in the context of defending adversarial examples.

In this work, we propose a quantization method that is realized by an extra intermediate layer, i.e., the quantization layer (Q-Layer). Our Q-Layer can be easily integrated into any existing architecture of CNN models. Specifically, the Q-Layer splits the mainstream of information that flows forward in a regular CNN model into two separate flows. Both flows share the same information passed by layers before the Q-Layer, but differ in the subsequent networks after the Q-Layer. These two flows produce two outputs, one is the quantized output, and the other is the Non-quantized output. Specifically, the non-quantized path is introduced to facilitate the gradient-based training, and to regularize the quantization operation. In the quantized path, we introduce non-differentiability to defend gradient-based attacks. It is important to note that, while gradient-based attacks cannot be directly applied to the quantized network, they can still be conducted by following the non-quantized path. Also, similar to most input transformation methods (Xu et al. (2018); Buckman et al. (2018)) proposed for defending adversarial examples, our quantization will inevitably lose some feature information, which might be useful for classification. In order to compensate for this loss of information, we further propose multi-head quantization, where we project data points to different sub-spaces and perform quantization within each sub-space. In particular, we perform the projection by re-weighting the input-channels of CNN with trainable parameters. This projection process can be interpreted as performing feature extraction from different points of view, hence help retain the overall effectiveness of our method without causing much performance degradation for the model to be protected. Last but not least, our proposed method can be readily combined with other existing defenses, e.g., adversarial training (Goodfellow et al. (2015)), to jointly improve the adversarial robustness of a protected CNN classifier.

In summary, we make the following contribution:

- We propose a quantization-based defense method for the adversarial example problem by designing a quantization Layer (Q-Layer) which can be integrated into existing architectures of CNN models. Our implementation is online available [1].
- We propose multi-head quantization to compensate for the possible information loss caused by the quantization process, and bring significant improvement to the adversarial robustness of an armed model under large perturbation.
- We evaluate our method under several representative attacks on MNIST and Fashion-MNIST datasets. Our experiment results demonstrate that the adoption of the Q-Layer can significantly enhance the robustness of a CNN against both black-box and white-box attack, and the robustness can be further improved by combining our method with adversarial training.

2 Related work

2.1 Adversarial attack

Given a neural network classifier N with parameters denoted by w, N can be regarded as a function that takes an input $x \in \mathbb{R}^{d_x}$ and produces an classification label y, i.e., $N(x; w) = y$ or $N(x) = y$ for notation simplicity. In principle, the goal of the adversarial attack is to create a perturbation $\delta \in \mathbb{R}^{d_x}$ to be added to a legitimate sample x for creating an adversarial example, i.e., $x + \delta$, which causes the target model N to produce a wrong classification result.

Depending on different threat models, adversarial attacks are categorized as black-box attacks or white-box attacks (Papernot et al. (2018)). Specifically, it is commonly assumed in the white-box attack scenario, that an attacker knows every detail of the target model. This dramatically eases the generation of impactful adversarial examples, and has stimulated researchers to propose various white-box attack methods, including the fast gradient sign method (FGSM) (Goodfellow et al.

[1] `https://anonymized`

(2015)), the basic iterative method (BIM) (Kurakin et al. (2017)), the Carlini-Wagner (CW) attack (Carlini & Wagner (2017)), and DeepFool (Moosavi-Dezfooli et al. (2016)). On the contrary, in the black-box attack scenario, an attacker is typically assumed to be restricted for accessing detailed information, e.g., the architecture, values of parameters, training datasets, of the target model. There have been many black-box attack methods proposed in prior work (Chen et al. (2017); Brendel et al. (2018); Papernot et al. (2016)). Representative black-box attacks typically exploit the transferability (Papernot et al. (2016)) of the adversarial examples, hence is also referred to as *transfer* black-box attacks. Explicitly, in transfer black-box attacks, an attacker can train and maintain a substitute model, then conduct white-box attacks on the substitute model to generate adversarial samples which retain a certain level of attack power to the target model. Since both black-box and white-box attacks rely on the white-box assumption, in the following, we mainly introduce several representative white-box attacks, namely the FGSM, BIM and CW attacks, which are also employed in our experiments due to their wide adoption as the benchmark attack methods (Samangouei et al. (2018a;b)).

Fast gradient sign method (FGSM) Goodfellow et al. (2015) proposed FGSM, in which δ is calculated by scaling the l_∞ norm of the gradient of the loss function L with respect to a legitimate input x as follows:

$$\delta = \epsilon \cdot sign(\nabla_x(L(N(x), y)))$$

where ϵ represents the maximally allowed scale of perturbation. This method represents a one-step approximation for the direction in the input space that affects the loss function most significantly.

Basic iterative method (BIM) Kurakin et al. (2017) proposed the BIM attack, which iteratively performs the FGSM hence generates more impactful adversarial examples at the expense of computational efficiency.

Carlini-Wagner (CW) attack Carlini & Wagner (2017) aimed to find the smallest perturbation to fool the target model, by solving the following optimization problem:

$$\begin{aligned} \min_{\delta} \quad & \|\delta\|_p + c \cdot L(x, \delta) \\ s.t. \quad & x + \delta \in [0, 1]^n \end{aligned}$$

where $c > 0$ is a tunable positive constant and $\|\|_p$ represents different norms. In our experiment, we consider l_∞ norm. L is designed to satisfy that $L(x, \delta) < 0$ if and only if $N(x + \delta) \neq N(x)$.

2.2 Defense methods

There have been many different defense methods (Xu et al. (2018); Goodfellow et al. (2015); Madry et al. (2018); Samangouei et al. (2018a;b); Buckman et al. (2018); Zhang & Liang (2019)) developed to battle with a large body of attacking methods. Here we briefly introduce three representative ones.

Adversarial training In the same work that introduced FGSM, Goodfellow et al. (2015) also proposed to train a target model with an augmented training set, which contains both original samples and adversarial samples generated by the target model itself. Considering the low computational cost needed by FGSM, it is usually used for generating target-specific adversarial examples in adversarial training. Madry et al. (2018) introduced a more general framework for generating adversarial examples and proposed to use iterative attacks, such as BIM, to produce adversarial samples for augmenting the training set. The resulting defense is regarded as among the most effective defense methods ().

Input quantization Xu et al. (2018) proposed feature squeezing, which transforms the input by reducing the range of pixel values then employs a filter to smooth the input image. Buckman et al. (2018) propose to encode the input with one-hot encoding or thermometer coding. Zhang & Liang (2019) add Gaussian noise to the input and cluster input pixels to perform quantization.

Input purification Samangouei et al. (2018b) propose PixelDefend, where they used a PixelCNN (van den Oord et al. (2016)) to model the distribution of input pixels, and differentiate adversarial samples from legitimate samples. Their results show that the pixel distribution of adversarial samples is indeed "unnatural" in comparison with legitimate samples. Then they proposed to purify input pixels to obtain a natural distribution and pass the purified input to the target model for further processing.

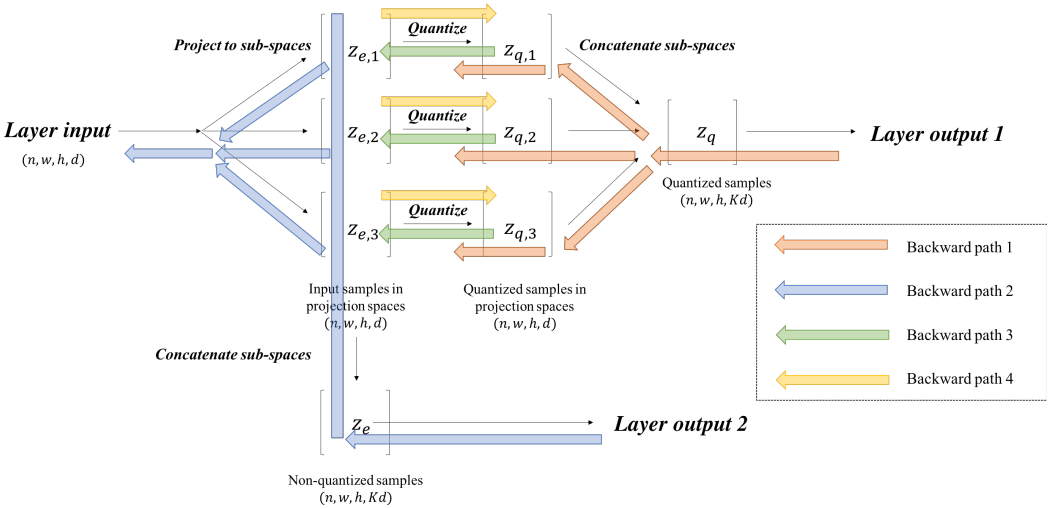


Figure 1: The overview of Q-Layer. The forward paths for the quantized output and the non-quantized output are drawn with solid black arrows and the four backward paths are drawn with red, blue, green and yellow arrows.

3 METHOD

3.1 OVERVIEW

Given a CNN which has the Q-Layer injected into the intermediate part (e.g., after the convolutional block and before the fully connected block or between two convolutional blocks) of the network, its input goes through the quantization path (Q-path), which consists of several steps of processing, including projection, quantization, and concatenation, then produce the quantized output for the subsequent network, as depicted in Figure 1. Note, that the quantization step forces pre-quantization data representation (outputs of the projection step), i.e., $z_{e,j}$ $(j = 1, 2, 3)$, to be replaced by the post-quantization representation, i.e., $z_{q,i}$, $(i = 1, 2, 3)$. Since the quantization step introduces non-differentiability in the Q-path, the backpropagation operation cannot be directly applied for this path. In order to make the enhanced model still trainable, we propose to concatenate the multi-head projection results and pass the concatenation to a separate subsequent network along the non-quantization path (E-path). It is natural to consider about having both paths connected to subsequent networks that are identical by sharing their weights, whereas, in our experiments, we observed inferior performance in comparison with having both paths connected to networks with different weights. Interested readers could refer to Appendix G for an ablation study which demonstrates this effect. Correspondingly, in Figure 1, we illustrate four backward paths that jointly contribute to updating the model. In particular, path 1, 3, and 4 constitute the backward path for the quantization operation, and path 2 represents the backward path for the non-quantization step.

3.2 SINGLE-HEAD QUANTIZATION

Given a neural network N, we can split it into two sub-network, denoted by N_F and N_B respectively, from somewhere in the middle of the network. Specifically, in this work, we split a regular CNN into an image encoder network which contains convolutional blocks and a fully-connected classifier network. Then given an input x, the output y of such a neural network is calculated by:

$$y = N(x) = N_B(N_F(x)). \tag{1}$$

We further assume the size of the intermediate output, i.e., $z_e = N_F(x)$, as $n \times w \times h \times d$, where n is the batch size, w, h are the width and height of the intermediate output, d is the number of channels. We consider z_e to be composed by $n * w * h$ panels, where an arbitrary panel j, i.e., z_e^j, is a vector of length d. As shown in Figure 1, the Q-Layer takes z_e as input panels, and outputs quantized panels z_q. Each panel of z_q represents a concept vector of dimension d. By setting the number of concepts

as some constant n_c a priori, we then have Q represented by a concept matrix of size $n_c \times d$. Let Q^i denote the i-th row of Q. Then the calculation in the Q-Layer is done by identifying the closest concept Q^{i^*} for each input panel z_e^j as shown in the following:

$$z_q^j = Q^{i^*}, \text{ where } i^* = \arg\min_i \left\| z_e^j - Q^i \right\| \tag{2}$$

After identifying the closest concept Q^{i^*} for z_e^j, we model the quantization operation as multiplying the entire concept matrix Q by an n_c-dimensional one-hot encoded identification vector I_{i^*}. As previously mentioned, we pass z_q and z_e to two different subsequent networks, denoted by N_Q and N_E respectively. Accordingly, we refer y_q and y_e as the outputs produced by the Q-path and the E-path, respectively. Then given an input x, the final output is as follows:

$$\begin{aligned} y_q &= N_Q(I(N_F(x))Q), \text{ where } I(z_e^j) = I_{i^*} = \begin{cases} 1, & \text{if } i^* = \arg\min_i \left\| z_e^j - Q^i \right\| \\ 0, & \text{otherwise} \end{cases}, \\ y_e &= N_E(N_F(x)). \end{aligned} \tag{3}$$

3.3 Multi-head quantization

Our quantization step introduced above essentially describes a built-in online clustering process. However, clustering in high dimensional space is challenging, mainly due to the existence of many irrelevant dimensions (Parsons et al. (2004)). Furthermore, in order to find rich concepts that reflect the similarity and difference between samples, our quantization step needs to locate clusters that may be embedded in different sub-spaces. As such, we introduce sub-space clustering (Parsons et al. (2004)) into our design and propose multi-head quantization to address these challenges as well as alleviate the possible information loss caused by the single-head quantization step.

More specifically, we first project an input representation to K sub-spaces, where K is a pre-defined hyper-parameter. Then we perform single-head quantization within each sub-space. At last, we concatenate the quantization results produced in all sub-spaces to obtain the final output. In the projection step, we apply a re-weight mechanism to d channels of an input representation. As mentioned before, we can consider a input batch as a $(n \times w \times h)$ panels, where a panel is represented as a d-dimensional vector containing features generated by d channels. Given the j-th panel $z_e^j \in \mathbb{R}^{1\times d}$, let $z_{e,i}^j$ be the i-th sub-space projection and $W_i \in \mathbb{R}^{d\times d}, b_i \in \mathbb{R}^{1\times d}$ be the projection parameters of the i-th sub-space, then we have:

$$z_{e,i}^j = z_e^j \odot \text{Softmax}(z_e^j W_i + b_i), \tag{4}$$

where $\odot$ denotes the Hadamard product. Denote the projection operation conducted in the i-th sub-space by P_i and the corresponding concept matrix in this sub-space by Q_i. Then the operation of concatenating the outputs from all sub-spaces is represented as follows:

$$C\Big(I\big(P_i(z_e)\big)Q_i, \forall i \in [1, K]\Big) = [I\big(P_1(z_e)\big)Q_1, \ldots, I\big(P_K(z_e)\big)Q_K]. \tag{5}$$

Given an input x, the formulations for y_q, y_e in the case of the multi-head quantization is as follows:

$$y_q = N_Q\left(C\Big(I\big(P_i(z_e)\big)Q_i, \forall i \in [1, K]\Big)\right), \quad y_e = N_E\Big(C\big(P_i(z_e), \forall i \in [1, K]\big)\Big) \tag{6}$$

3.4 Optimization Loss

By following the design of VQ-VAE (van den Oord et al. (2017)) and SOM-VAE (Fortuin et al. (2019)), we decompose the quantization loss into two separate loss terms and finally specify the following four terms (as shown in Figure 1) in the final training loss, i.e.,

$$L = c_1 \cdot L_Q + c_2 \cdot L_E + \alpha \cdot L_{Q\to E} + \beta \cdot L_{E\to Q}, \tag{7}$$

where c_1, c_2, α, β are tunable hyper-parameters. Specifically, the first loss L_Q represents the standard cross-entropy loss caused by the quantized output. This loss is specified to optimize the weight

parameters of N_Q and the Q-Layer to fit the classification task. As previously mentioned, the Q-path does not allow direct backpropagation, as such, in order to optimize N_F, we set the second cross-entropy loss, i.e., L_E, to train N_F and N_E by following the E-path.

The last two loss terms are specified for optimizing the quantization performance by directly evaluating the distance between concept vectors, which can be regarded as clustering centroids, and learned data projections. In particular, the third loss $L_{Q\to E}$ measures the l_2 distance between z_q and a fixed z_e. In other words, we apply the "stop-gradient" operation (denoted as sg) to z_e to ensure that this loss only moves z_q to z_e instead of the opposite.

$$L_{Q\to E} = ||z_q - sg(z_e)||_2^2, \quad L_{E\to Q} = ||sg(z_q) - z_e||_2^2 \tag{8}$$

This is a direct analogy to clustering as by minimizing $L_{Q\to E}$, all row vectors of the parameter matrix Q keep moving towards different clustering centers. Similarly, the fourth loss, i.e., $L_{E\to Q}$, measures the "commitment" of data projections to their corresponding clustering centroids. Here we apply the stop-gradient operation to z_q while minimizing this loss. By optimizing $L_{E\to Q}$, we try to shape the learned projections of samples to be more similar to their corresponding concepts, hence eventually forming densely distributed clusters. Although we have followed prior work (van den Oord et al. (2017)) to decompose the quantization loss as described above and allowed more flexible control over $L_{Q\to E}$ and $L_{E\to Q}$ by tuning α and β, however, in our experiments, after tuning α and β with cross-validation, we observed the ratio of these two hyper-parameters, if within certain range (for example, 0.1 to 10), had insignificant influence on the final performance. A similar effect has also been reported by van den Oord et al. (2017). As such, in Section 4, we mainly present the results obtained by setting $\alpha = \beta$.

3.5 Update for Inactive concepts

If the concept matrix is not properly initialized or optimized, a concept which is far away from all data projections may remain constant during training. We refer to concepts that behave as described above as "inactive" concepts and refer to other concepts that can be effectively updated as "active" concepts. This issue may severely impact the final performance when there are relatively many inactive concepts.

To update inactive concepts, we design two updating strategies. One is to force an inactive concept to move its closest panel directly. The other is to re-initialize an inactive concept as its closest panel. In our experiments, we mainly use the first strategy due to its better effectiveness. Specifically, the first strategy is implemented by adding a special loss term, which calculates the distance between each inactive concept to its closest panel. We then optimize this loss after each training epoch.

4 Evaluation

4.1 Experiment Setup

In our experiments, we considered two types of attacks, namely the black-box and the white-box attacks. Under each type of attacks, we evaluated the effectiveness of our proposed Q-Layer by comparing both the accuracy and robustness of two standard CNNs built with and without the Q-Layer obtained on a clean testing set and an adversarial testing set constructed from this clean testing set. The accuracy achieved by a CNN on the clean testing set reflects its standard generalization performance, while its robustness is represented by the accuracy obtained by this CNN on the adversarial testing set. In particular, under white-box attacks, for a CNN equipped with the Q-layer, the direct gradient-based attack can only be applied to the E-path, as mentioned previously. We refer to this type of attack as the E-path white-box attack. Furthermore, we assumed that an attacker might attack the non-differentiable Q-path by ignoring the quantization operation and directly concatenate z_e to N_Q to build a substitute attack path following equation 9, which we refer as the Q-path white-box attack.

$$\nabla_x y_q = \nabla_{z_q} N_Q(z_q) \nabla_x z_q \approx \nabla_{z_e} N_Q(z_e) \nabla_x z_e \tag{9}$$

Moreover, under both types of attacks, we also evaluated CNNs built with and without the Q-Layer by considering the case where both models were enhanced by adversarial training Goodfellow et al.

(2015). The models considered in this case are referred to as *adversarial-trained* models. Correspondingly, the models trained without adversarial training are referred to as *raw-trained* models. Note, in the following experiments, we only focus on the classification accuracy obtained by the Q-path by CNNs armed by the Q-Layer. We report the accuracy obtained by the E-path in Appendix D.

4.1.1 DATASET

The datasets adopted in our experiments include the MNIST (LeCun & Cortes (2010)) and the Fashion-MNIST (Xiao et al. (2017)) datasets [2]. For each dataset, we had five subsets of samples, including a training set, an adversarial training set, a validation set, a clean testing set, and an adversarial testing set. A clean validation set was first constructed by randomly selecting 10,000 samples from the original training set, hence leaving the final training set contained the rest 50,000 training samples. Then we created adversarial examples from this clean validation set by using a pre-trained CNN_A[3] and the FGSM method. This pre-trained CNN_A only acted as a source model for creating adversarial validation samples and was not involved in either the training or testing process. The clean validation set was then mixed with its adversarial counterpart to build the final validation set. This mixed validation set was used for selecting models that achieved the best performance. Through this, we attempted to avoid the case of selecting models that can only perform well with clean samples. Instead, we tried to evaluate the performance of models that are naturally robust against adversarial examples. A comparison between the results obtained following the above scheme and the results obtained by models selected by using the clean validation set is provided in Appendix C.

During the testing phase, depending on different attack scenarios, we constructed different adversarial testing sets. Specifically, under black-box attacks, we independently trained two source models for generating two sets of adversarial testing samples. One of the source models had the same structure as the CNN_A but was trained with a different random initialization. The other source model, denoted by CNN_B, had a structure that was different from that of CNN_A. Both source models were used to simulate the scenario where an attacker generated adversarial examples from some unknown models. Under the white-box attacks, for each target model, its associated adversarial testing set was generated by directly attacking this model.

Recall that we can apply our method in tandem with adversarial training. Therefore, on the MNIST, we created adversarial examples for adversarial training by setting $\epsilon = 0.2$. For the mixed validation and adversarial testing set, we created adversarial examples by setting $\epsilon = 0.2/0.3$. This setup is due to the observation that FGSM-based adversarial training is less effective against adversarial samples created with a larger scale of perturbation. As a result, we attempted to simulate a more critical scenario in this setup. On the Fashion-MNIST, we noticed that by setting $\epsilon = 0.2/0.3$, the resulting adversarial images were severely distorted. As such, we created adversarial examples by setting $\epsilon = 0.1/0.2$.

4.1.2 MODEL SETUP

In our experiments, we compared the robustness of three target models: CNN_A, CNN with a basic Q-Layer ($K = 1, n_c = 64$, denoted as Q-base), and CNN with a large Q-Layer ($K = 4, n_c = 64$, denoted as Q-large). Q-base and Q-large share the same architecture with CNN_A, but with an additional Q-Layer after the convolution blocks and before the fully-connect layer. We have also evaluated the case where the Q-Layer was inserted between two convolution blocks of CNN_A. The results are showed in Section 4.4.

Besides, we provided the architectures of the target and substitute models, the hyper-parameters, and other implementation details in Appendix A/B.

[2] Both MNIST and Fashion-MNIST contain 60,000 training samples and 10,000 testing samples. All samples are in grey-scale with the size of 28×28.

[3] CNN_A denotes a CNN with the type-A architecture, which is same as the architecture adopted by Samangouei et al. (2018a), as shown in the Appendix A

	Clean	FGSM		BIM	
	None	0.2	0.3	0.2	0.3
CNN	**98.73**	75.77	40.03	62.52	16.02
Q-base	96.57	84.86	52.69	84.08	38.22
Q-large	97.38	**85.87**	**65.19**	**86.48**	**67.01**
CNN + adv	**98.73**	**96.78**	80.57	96.18	64.60
Q-base + adv	98.58	96.36	**91.7**	**96.72**	**92.46**
Q-large + adv	98.33	95.83	84.58	96.11	86.4

(a) Results for black-box attacks on MNIST

	Clean	FGSM		BIM	
	None	0.1	0.2	0.1	0.2
CNN	**86.65**	66.15	43.07	63.94	38.7
Q-base	84.76	**72.24**	47.77	**72.67**	**52.41**
Q-large	82.3	69.35	**51.92**	70.24	52.32
CNN + adv	**88.25**	**83.56**	57.81	**83.68**	51.76
Q-base + adv	83.21	79.09	**71.07**	79.53	**73.12**
Q-large + adv	82.32	78.78	70.31	78.89	70.84

(b) Results for black-box attacks on Fashion-MNIST

Table 1: Comparison of classification accuracy for different target models on the MNIST under black-box attack with source model CNN_A. We use "N+adv" to refer adversarial-trained models.

4.2 Results for Black-box attacks

As previously mentioned in Section 4.1.1, we use two substitute models, i.e., CNN_A and CNN_B, to implement black-box attacks for a target model. Notably, in our experiments, we observed that the attacks conducted by using CNN_A were generally more impactful than the attacks conducted by using CNN_B. As such, due to space limitations, here we only demonstrate the results obtained under attacks performed by CNN_A and provide the results obtained for CNN_B in Appendix E. Also, we have adopted three representative attacks, namely the FGSM attack, the BIM attack, and the CW attack (introduced in Section 2). In Table 1a and Table 1b, we only present the results obtained under the FGSM and BIM attacks, as we have observed that, despite being a powerful attack under the white-box scenario, the CW attack produced less effective black-box attacks.

MNIST From Table 1a, we can observe that, by inserting the Q-Layer into the CNN, its robustness is improved significantly. Take the results for raw-trained Q-large as an example, in comparison with the unarmed CNN, the accuracy under the FGSM attack of $\epsilon = 0.3$ rises from 40.03 to 65.19, and the accuracy under BIM attack of $\epsilon = 0.3$ rises from 16.02 to 67.01, which is even higher than the accuracy obtained by the adversarial-trained CNN.

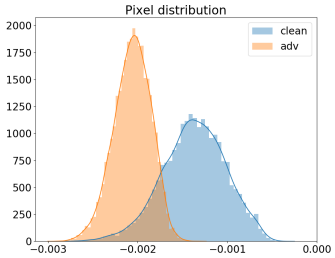
With adversarial training, the robustness of all models increases. In particular, while these models have similar robustness under small-scale perturbations, as the scale of perturbation increases, the robustness of the target CNN decreases more significantly than Q-large and Q-base, especially for the BIM attack.

Fashion-MNIST In Table 1b[4], we show the results obtained under black-box attacks for Fashion-MNIST. Similar to MNIST, we can observe improved robustness for raw-trained Q-base and Q-large, in comparison with CNN. Also, after adversarial training, the robustness of the three models are close under small perturbation attack, while Q-base and Q-large outperform CNN under large perturbation.

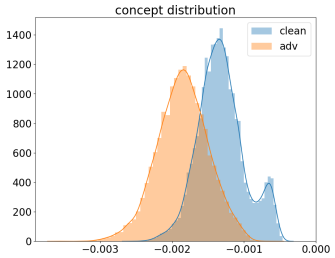
In addition to the results shown above, we have also performed several sets of ablation study for exploring the influence of the number of sub-spaces and the number of concepts on black-box robustness and presented the results in Appendix F.

Visualizing the distributions of pixels and concepts Inspired by PixelDefend (Samangouei et al. (2018b)), which proposed to use a PixelCNN van den Oord et al. (2016) to model and compare the distribution of pixels in clean and adversarial samples, here we use a PixelCNN to model and compare the distribution of concepts learned from clean and adversarial samples. As shown in Figure 2a/2b, for the MNIST dataset and under the FGSM attack, the distribution of pixels exhibits much more significant distortion than the distribution of learned concepts. This result clearly indicates that the quantization effectively blocks the adversarial perturbations, hence resulting in improved robustness.

[4]We've noted that the adversarial-trained CNN reaches a higher classification accuracy on clean samples, which is due to our model selection strategy. When selected on the clean validation set, the classification accuracy of the adversarial-trained CNN would lower than that of the raw-trained CNN in most cases.



(a) The pixel distribution on MNIST



(b) The concept distribution on MNIST

Figure 2: Visualization of the pixel and concept distributions for MNIST. We use FGSM attack with $\epsilon = 0.4$ to generate adversarial samples for better visualization. For this case, the KL divergence is 5.39 for the pixel distribution and 4.33 for the concept distribution.

	Clean	FGSM		BIM		CW	
	None	0.2	0.3	0.2	0.3	0.2	0.3
CNN	**98.73**	43.18	7.84	6.69	0.82	58.00	39.95
Q-base	96.57	**77.23**	**58.47**	59.05	**55.33**	**90.46**	**85.77**
Q-large	97.38	65.99	30.68	**60.24**	16.95	87.64	80.03
CNN + adv	**98.73**	**95.63**	90.23	88.03	7.77	95.65	90.08
Q-base + adv	98.58	95.21	**91.28**	**94.64**	**83.07**	**97.37**	94.85
Q-large + adv	98.33	95.34	90.83	94.11	79.94	97.34	**95.3**

(a) Results for white-box attacks on MNIST

	Clean	FGSM		BIM		CW	
	None	0.1	0.2	0.1	0.2	0.1	0.2
CNN	**86.65**	32.16	11.28	20.29	8.06	44.01	20.95
Q-base	84.76	**53.30**	**24.17**	**50.77**	14.36	**74.87**	**67.82**
Q-large	82.30	44.01	17.33	42.97	**14.99**	66.48	56.92
CNN + adv	**88.25**	**81.39**	**62.84**	65.41	8.53	78.62	57.60
Q-base + adv	83.21	76.57	59.27	**75.13**	41.85	**80.77**	**73.4**
Q-large + adv	80.04	75.36	58.79	74.04	**53.71**	78.82	69.43

(b) Results for white-box attacks on Fashion-MNIST

Table 2: Comparison of classification accuracy for different target models on the MNIST under white-box attack.

4.3 Results for White-box Attack

As mentioned in Section 4.1, in order to conduct attacks even when the Q-path does not suffer from gradient-based attacks, we assume that an attacker might use a shortcut from z_e to N_Q to perform attacks. However, we observed in our experiments (shown in Appendix H) that this type of attack was generally weaker than the E-path attack. Therefore, here we only present results obtained by the E-path attack in Table 2a and Table 2b.

As shown in Table 2a, raw-trained Q-large and Q-base significantly outperform CNN. For attacks with small perturbations, adversarial trained CNN performs slightly better than Q-large and Q-base. However, as the scale of perturbation increases, Q-large and Q-base restore their superior robustness. Similar results can also be observed for Fashion-MNIST, as shown in Table 2b. In Section 4.4, we also show that inserting the Q-Layer between two convolution blocks brings even stronger robustness against white-box attack.

4.4 Inserting Q-Layer between two convolution blocks

In the following experiments, we show the flexibility of varying the position where the Q-Layer is inserted. In particular, we refer to a CNN which has a Q-Layer inserted between its two convolution blocks as Q-inner. The Q-inner model has the same parameter setting as the Q-large ($K = 4, n_c = 64$) model used in the previous experiments.

	Clean	FGSM		BIM	
	None	0.2	0.3	0.2	0.3
CNN	**98.73**	75.77	40.03	62.52	16.02
Q-large	97.38	85.87	**65.19**	86.48	**67.01**
Q-inner	98.09	**86.51**	61.60	**87.62**	63.49

(a) Results for black-box attacks with source model CNN_A

	Clean	FGSM		BIM		CW	
	None	0.2	0.3	0.2	0.3	0.2	0.3
CNN	**98.73**	43.18	7.84	6.69	0.82	58.00	39.95
Q-large	97.38	65.99	30.68	60.24	16.95	87.64	**80.03**
Q-inner	98.09	**82.41**	**67.15**	**83.08**	**81.96**	**90.36**	79.46

(b) Results for white-box attacks

Table 3: Comparison of classification accuracy for Q-large and Q-inner on the MNIST, under black-box attack and white-box attacks.

In Table 3a and Table 3b, we demonstrate the comparison results under both black-box and white-box attacks on the MNIST dataset. Under black-box attacks, we observe that Q-inner and Q-large achieve comparable robustness. However, under the white-box attacks, it can be seen that Q-inner performs much better than Q-large. These results not only demonstrate the flexibility of the Q-Layer, but also indicate that applying the quantization at an early stage of the information propagation can be more effective for filtering out an adversarial perturbation, of which the scale has not been sufficiently amplified as the perturbation propagates to the deeper parts of the network.

5 Conclusion

In this paper, we have designed and implemented a quantization layer (Q-Layer) to protection CNN classifiers from the adversarial attacks, and presented the experiment results which show that, by simply inserting one Q-Layer into a regular CNN, its adversarial robustness under both white-box and black-box attacks obtains significant improvement. Moreover, we have combined our method in tandem with adversarial training. The empirical results show that the Q-layer can make a CNN benefit more from adversarial training and even perform well under attacks with larger perturbations. One limitation of this work is due to the uncertainty introduced by the random initialization of concept matrix. This issue also exists in many other clustering algorithms. In this work, we alleviate the impact of this issue by reactivating inactivate concepts. Future work would pursue other approaches on constructing the concept matrix, e.g., regularizing the concept matrix with specific semantic constrains, and using the E-path as a learned index to retrieve information stored in the concept matrix, which acts as an external memory.

References

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, pp. 3319–3327. IEEE Computer Society, 2017.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR (Poster)*. OpenReview.net, 2018.

Jacob Buckman, Aurko Roy, Colin Raffel, and Ian J. Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *ICLR (Poster)*. OpenReview.net, 2018.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pp. 39–57. IEEE Computer Society, 2017.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *AISec@CCS*, pp. 15–26. ACM, 2017.

Vincent Fortuin, Matthias Hüser, Francesco Locatello, Heiko Strathmann, and Gunnar Rätsch. SOM-VAE: interpretable discrete representation learning on time series. In *ICLR (Poster)*. OpenReview.net, 2019.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR (Poster)*, 2015.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR (Workshop)*. OpenReview.net, 2017.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *CoRR*, abs/1908.03265, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. 2018.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *CVPR*, pp. 2574–2582. IEEE Computer Society, 2016.

Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian M Molloy, et al. Adversarial robustness toolbox v0. 4.0. *CoRR*, abs/1807.01069, 2018.

Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016.

Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. Sok: Security and privacy in machine learning. In *EuroS&P*, pp. 399–414. IEEE, 2018.

Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations*, 6(1):90–105, 2004.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018a.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *ICLR (Poster)*. OpenReview.net, 2018b.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR (Poster)*, 2014.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *NeuraIPS*, pp. 4790–4798, 2016.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeuraIPS*, pp. 6306–6315, 2017.

Jianyu Wang, Zhishuai Zhang, Vittal Premachandran, and Alan L. Yuille. Discovering internal representations from object-cnns using population encoding. *CoRR*, abs/1511.06855, 2015.

Jianyu Wang, Zhishuai Zhang, Cihang Xie, Yuyin Zhou, Vittal Premachandran, Jun Zhu, Lingxi Xie, and Alan L. Yuille. Visual concepts and compositional voting. *CoRR*, abs/1711.04451, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *NDSS*. The Internet Society, 2018.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV (1)*, volume 8689 of *Lecture Notes in Computer Science*, pp. 818–833. Springer, 2014.

Yuchen Zhang and Percy Liang. Defending against whitebox adversarial attacks via randomized discretization. In *AISTATS*, volume 89 of *Proceedings of Machine Learning Research*, pp. 684–693. PMLR, 2019.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.

A NEURAL NETWORK ARCHITECTURES

Model A,B are two neural network architectures used in Samangouei et al. (2018a) for the MNIST and Fashion-MNIST datasets. In this paper, A is used for both target CNN and substitute model. B is only used for substitute model.

A	B
Conv(64, 5×5, 1) + ReLU	Conv(128, 3×3, 1) + ReLU
Conv(64, 5×5, 2) + ReLU	Conv(64, 3×3, 2) + ReLU
Dropout(0.25)	Dropout(0.25)
FC(128) + ReLU	FC(128) + ReLU
Dropout(0.5)	Dropout(0.5)
FC(10) + Softmax	FC(10) + Softmax

Table 4: Neural network architectures used for target models and substitute models.

B IMPLEMENTATION DETAILS

When train CNN with a Q-Layer, we use RAdamOptimizer (Liu et al. (2019)) and set learning rate as 0.001. As for losses, we set $c_1 = 1, c_2 = 1, \alpha = 0.001, \beta = 0.001$. The scale of α and β are chosen to make $\alpha \cdot L_{Q \to E}$ and $\beta \cdot L_{E \to Q}$ have smaller but close magnitude of L_Q and L_E. The ratio of c_1 and c_2 do not have significant influence on the results, for they are optimizing different part of the network. In our experiments, We didn't find significant change of results when we tuned the ratio of α and β. As for inactive concepts updating strategy one, i.e., optimizing the distance between inactive concepts to its closest data point, we use AdamOptimizer and set learning rate as 0.001. Specifically, when training Q-inner on MNIST, we set $\alpha = 0.0001, \beta = 0.0001$ and use the second inactive concept updating strategy (otherwise it could not converge), keeping other hyper-parameters unchanged. All the hyper-parameter tuning are based on train set and validation set. We train each model three times with different random initialization and select the model with highest validation accuracy as the final model to test.

We use Adversarial Robustness 360 Toolbox (Nicolae et al. (2018)) to generate adversarial samples. For each attack, we adjust ϵ, set batch size to be 128, leaving other hyper-paramters as default settings in Adversarial Robustness 360 Toolbox. Additionally, when implementing BIM attack, we set ϵ step to be 0.01.

C COMPARISON OF SELECTING MODEL FROM THE CLEAN VALIDATION SET AND FROM THE MIXED VALIDATION SET

In this section, we compare two CNN models on MNIST, all share the same structure with CNN_A, independently trained, but are selected on the clean validation set and mixed validation set, respectively.

	Clean	FGSM		BIM	
	None	0.2	0.3	0.2	0.3
CNN, clean-selected	**99.48**	55.71	22.29	17.66	0.63
CNN, mixed-selected	98.73	**75.77**	**40.03**	**62.52**	**16.02**

Table 5: Comparison of classification accuracy for clean-selected CNN and mix-selected CNN on the MNIST under black-box attack with source model CNN_A.

As shown in Table 5, by selecting models on the mixed validation set, we get a much more robust model, with slight sacrificing on clean accuracy.

	Clean	FGSM		BIM	
	None	0.2	0.3	0.2	0.3
Q-base	96.57/97.46	84.86/77.64	52.69/37.65	84.08/69.06	38.22/17.13
Q-large	97.38/**98.59**	**85.87**/70.52	**65.19**/34.44	**86.48**/72.86	**67.01**/29.32
Q-base + adv	98.58/98.84	96.36/96.84	**91.7**/86.0	96.72/96.88	**92.46**/81.51
Q-large + adv	98.33/**99.09**	95.83/**97.15**	84.58/64.42	96.11/**97.36**	86.4/60.9

(a) Results for black-box attacks on MNIST with source model CNN_A

	Clean	FGSM		BIM		CW	
	None	0.2	0.3	0.2	0.3	0.2	0.3
Q-base	96.57/97.46	**77.23**/46.91	**58.47**/44.09	59.05/43.69	**55.33**/43.69	**90.46**/55.84	**85.77**/32.31
Q-large	97.38/**98.59**	65.99/8.54	30.68/2.13	**60.24**/2.16	16.95/1.98	87.64/49.59	80.03/32.07
Q-base + adv	98.58/98.84	95.21/94.6	**91.28**/90.62	**94.64**/87.83	**83.07**/14.62	**97.37**/94.32	94.85/86.53
Q-large + adv	98.33/**99.09**	95.34/**95.99**	90.83/88.28	94.11/91.91	79.94/34.34	97.34/95.03	**95.3**/88.12

(b) Results for white-box attacks on MNIST

Table 6: Comparison of Q-path classification accuracy and E-path classification accuracy on the MNIST under black-box attack and white-box attack. The two scores separated by "/" represent the accuracy obtained by following the Q-path and E-path, respectively. We use "+adv" to refer adversarial-trained models.

	Clean	FGSM		BIM	
	None	0.2	0.3	0.2	0.3
CNN	**98.73**	82.49	51.60	73.14	28.89
Q-base	96.57	**88.98**	64.57	**89.25**	57.83
Q-large	97.38	86.73	**68.48**	88.51	**74.05**
CNN + adv	**98.73**	**97.29**	82.79	96.78	71.56
Q-base + adv	98.58	96.88	**93.45**	**96.88**	**93.83**
Q-large + adv	98.33	96.36	86.72	96.41	89.32

(a) Results for black-box attacks on MNIST

	Clean	FGSM		BIM	
	None	0.1	0.2	0.1	0.2
CNN	**86.65**	72.62	52.03	70.87	49.70
Q-base	84.76	**74.64**	54.55	**75.85**	58.33
Q-large	82.3	71.95	**57.33**	73.16	**61.28**
CNN + adv	**88.25**	**84.19**	64.84	**83.70**	63.29
Q-base + adv	83.21	80.07	**73.54**	80.12	71.63
Q-large + adv	82.32	79.13	72.10	79.41	**74.42**

(b) Results for black-box attacks on Fashion-MNIST

Table 7: Comparison of classification accuracy for different target models on the MNIST under black-box attack with source model CNN_B. We use "+adv" to refer adversarial-trained models.

D Comparison of Q-path classification accuracy and E-path classification accuracy

In this section, take MNIST as an example, we compare the Q-path classification accuracy and E-path classification accuracy under black-box attack and while-box attack. In the following, we've reported two classification accuracy scores for each of the Q-base and Q-large models. One score represents the accuracy obtained by following the Q-path, and the other represents the accuracy obtained by following the E-path.

As shown in Table 6a and Table 6b, for both raw-trained Q-base and Q-large, the accuracy obtained by the Q-path is significantly higher than that obtained by the E-path, especially when ϵ is large, which prove the efficiency of quantization. We also note that in certain cases, after adversarial training, for Q-large and Q-base under small perturbations, the accuracy obtained by the Q-path becomes slightly lower than that obtained by the E-path. Recall that the E-path represents the passage in a regular CNN. Therefore, this difference between the Q-path and E-path may be due to the reason that the quantization inevitably causes loss of information, hence leading to a lower accuracy score for the Q-path.

E Black-box attack with source model CNN_B

In this appendix, we compare the classification results of different target models on MNIST and Fashion-MNIST, under black-box attack with source model CNN_B, whose architecture is different from the target CNN. The results on MNIST is showed in Table 7a. The results on Fashion-MNIST is showed in Table 7b. Same as the observation in Table 1a and Table 1b, inserting a Q-Layer improves the robustness of the network.

	$n_c = 16$	$n_c = 64$	$n_c = 128$
$K = 1$	46.69	52.69	44.64
$K = 2$	60.38	60.42	49.42
$K = 4$	**65.23**	**65.19**	**60.68**

(a) Results before adversarial training

	$n_c = 16$	$n_c = 64$	$n_c = 128$
$K = 1$	70.89	**91.7**	83.77
$K = 2$	80.08	85.5	**85.74**
$K = 4$	**82.28**	84.58	83.65

(b) Results after adversarial training

Table 8: The comparison of Q-path accuracy when inserting Q-Layer with different K and n_c, under FGSM black-box attack with CNN_A, $\epsilon = 0.3$.

F ABLATION STUDY ON THE NUMBER OF SUB-SPACES AND THE NUMBER OF CONCEPTS

To further study the effect of K (the number of sub-spaces) and n_c (the number of concepts), we insert Q-layer with $K = 1, 2, 4$ and $n_c = 16, 64, 128$ to CNN and compare its robustness on MNIST under FGSM black-box attack with $\epsilon = 0.3$.

As shown in Table 8a, before adversarial training, more sub-spaces bring stronger robustness. After adversarial training, when $n_c = 16$, models with larger K demonstrate higher accuracy; however, when $n_c = 64/128$, the relationship of K and accuracy is not clear. We speculate that when $n_c = 16$, quantizing features causes too much loss in information, therefore additional information introduced by adding more sub-spaces significantly improves accuracy. However, when n_c is large enough to represent different concepts, the importance of K decreases after adversarial training.

G ABLATION STUDY FOR THE SEPARATED SUBSEQUENT NETWORKS AND SHARED SUBSEQUENT NETWORKS

In the shared subsequent network setting, we let N_Q and N_E share parameters, while in the separated subsequent network setting, N_Q and N_E do not. We compare the robustness of two independent Q-large models with separated subsequent networks and with shared subsequent network. We report the black-box attack results on MNIST in Table 9 and white-box attack results in Table 10. The results proves that, the separated subsequent networks do help robustness under both black-box attack and white-box attack. Nonetheless, separated subsequent networks are harder to optimize, thus we recommend users to use separated subsequent networks if they are not very deep, and use shared subsequent networks otherwise.

	Clean	FGSM		BIM	
	None	0.2	0.3	0.2	0.3
Q-large	97.38	**85.87**	**65.19**	**86.48**	**67.01**
Q-large, share	**98.44**	84.43	48.64	84.52	47.19

Table 9: Classification accuracy comparison of Q-large with separated subsequent networks and shared subsequent networks on MNIST, under black-box attack with source model CNN_A.

	Clean	FGSM		BIM		CW	
	None	0.2	0.3	0.2	0.3	0.2	0.3
Q-large	97.38	**65.99**	**30.68**	**60.24**	**16.95**	**87.64**	**80.03**
Q-large, share	**98.44**	56.96	16.10	34.69	8.17	87.18	79.48

Table 10: Classification accuracy comparison of Q-large with separated subsequent networks and shared subsequent networks on MNIST, under white-box attack.

H WHITE-BOX ATTACK ON Q-PATH

As mentioned before, the original Q-path for Q-layer could not be attacked with gradient-based method. A clever attacker might build a shortcut path from z_e to N_Q to attack, which we refer as Q-path white-box attack. However, due to the differences between N_Q and N_E, attack through

short-cut path is usually weak. In this section, we show the Q-path white-box attack results and E-path white-box attack results in Table 11. We could observe that, at most of the time, attacking Q-path reduces less accuracy than attack E-path.

	Clean	FGSM		BIM		CW	
	None	0.2	0.3	0.2	0.3	0.2	0.3
CNN	98.73	43.18	7.84	6.69	0.82	58.00	39.95
Q-base, E-path	96.57	77.23	58.47	59.05	55.33	90.46	85.77
Q-base, Q-path	96.57	85.93	63.12	68.9	60.78	97.29	96.58
Q-large, E-path	97.38	65.99	30.68	60.24	16.95	87.64	80.03
Q-large, Q-path	97.38	70.36	36.29	62.62	19.03	92.72	87.13
CNN + adv	98.73	95.63	90.23	88.03	7.77	95.65	90.08
Q-base + adv, E-path	98.58	95.21	91.28	94.64	83.07	97.37	94.85
Q-base + adv, Q-path	98.58	97.05	94.25	96.28	86.73	98.34	97.8
Q-large + adv, E-path	98.33	95.34	90.83	94.11	79.94	97.34	95.3
Q-large + adv, Q-path	98.33	95.74	<u>90.31</u>	94.11	<u>73.77</u>	97.8	96.47

Table 11: Classification accuracy comparison of different target models on MNIST, under Q-path and E-path white-box attack. We underline the results when Q-path attack is stronger than E-path attack.