A. Implementation Details

In this section, we present the implementation details for our method. Given the pre-trained decoder, the detailed network architecture is described in Fig. A.1, which is based on classifier-decoder structure. We employ the existing backbone networks (i.e., ResNet50 [17]) as a target classifier. Note that the classifier of QualNet reveals a subtle difference from original ResNet; we remove max-pooling and replace the first convolution of stride 1 behind max-pooling with the convolution of stride 2, which produces slightly better performance on corrupted images. We use the modified classifier as default. We use i-RevNet300 [25] as an invertible decoder, whose performance is comparable to that of ResNet50 (released code available on https://github.com/jhjacobsen/pytorch-i-revnet). Furthermore, the reconstructed image by (4) is clipped by a certain lower and upper bound (set from 0 to 1) for stable training. Note that the invertible decoder is trained at the first stage and it is not updated at the second stage. The structure of i-RevNet is similar to ResNet, which consists of some reversible blocks [15], global average pooling (GAP) [31] and softmax layer. In a reversible block, each layer's activations can be reconstructed from the next layer's activations (bijective property). Furthermore, the pooling or stride 2 operation is done by rearranging pixel elements from spatial to channel, which is an invertible down-sampling and is widely used as a component of sub-pixel convolution in super-resolution tasks [40]. Since the i-RevNet guarantees the invertible property only in a series of reversible blocks (from input to final feature map before GAP), we discard a global average pooling and softmax layer after its training and use its inversion as an invertible decoder. To connect an classifier to an invertible decoder, we introduce an additional 3x3 convolution (without batch normalization [24] and ReLU [35]) as shown in Fig. A.1. Furthermore, once we obtain an invertible decoder, it can be attached to any backbone networks. Hence, no more training for an invertible decoder is required to connect with other classifier architectures. For data augmentation at the second stage, we use 15 corruption functions used in ImageNet-C [19] during the training (released code available on https://github.com/hendrycks/robustness/blob/master/ImageNet-C/imagenet_c/imagenet_c/corruptions.py).

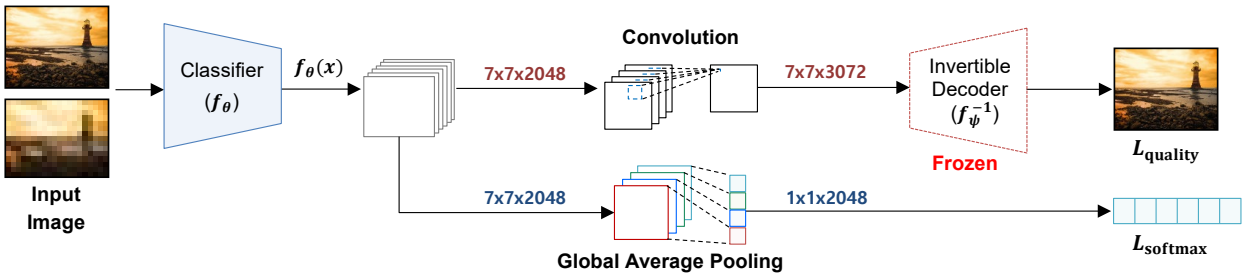

Figure A.1. Network architecture overview. The network architecture consists of two components: classifier and decoder. We use the existing backbone networks (e.g., ResNet50 [17]) as a classifier. The i-RevNet300 [25] is employed as an invertible decoder. The decoder takes the final feature map of the classifier as an input, which outputs reconstructed images.

B. Performance on individual corruption types

We investigate our method to show the performance on individual corruption types. We train ResNeXt101-32x8d [48] with ImageNet-1K. We choose a vanilla model (trained with clean images only) and naïve data augmentation model (trained with clean and corrupted images) as baselines. As reported in Table B.1, our method achieves consistent performance improvement on individual corruption types except for snow corruption type, compared with the naïve data augmentation model. Furthermore, our method also improves the clean accuracy even though there is a performance trade-off between clean and corruption accuracy.

Methods	Accuracy (%)																
	Clean	Corrupt. Average	Blur				Noise			Digital				Weather			
			Motion	Glass	Zoom	Defocus	Gaussian	Shot	Impulse	Contrast	Elastic	Pixelate	JPEG	Snow	Frost	Fog	Bright
Vanilla	79.64	44.64	44.59	28.29	42.17	42.87	39.12	36.61	34.80	41.97	48.64	53.98	59.68	36.49	41.82	48.87	69.66
Naïve Augmentation	79.54	65.93	68.86	56.21	69.32	59.88	60.05	59.39	60.09	66.08	69.47	71.82	68.16	**69.08**	66.29	69.42	74.90
QualNeXt101 (ours)	**79.84**	**66.62**	**69.65**	**58.61**	**69.93**	**60.21**	**61.18**	**60.06**	**60.92**	**67.41**	**70.23**	**72.62**	**68.18**	68.38	**66.63**	**69.62**	**75.71**

Table B.1. Accuracy (%) on individual corruption types. We train ResNeXt101-32x8d [48] with ImageNet-1K [5] for all methods. We evaluate ImageNet-1K validation set and ImageNet-C [19]. "Clean" indicates Top-1 clean accuracy (%) and "Corrupt. Average" indicates average accuracy (%) over 15 corruption types. The other types show the individual corruption average accuracy (%) over their five severity levels. The best results are indicated in bold.

C. Additional ablation study

C.1. Quality losses

It is crucial to study which loss is more effective for our quality loss (4). We experiment with five types of losses: L1 loss, L2 loss, likelihood loss [28], adversarial loss [16] and perceptual loss [26]. To calculate the gaussian likelihood loss, the mean and standard deviation images are required. To this end, we add two convolutions at the end of the decoder as in [28]. We also apply the adversarial loss to our method by introducing a discriminator. The perceptual loss is implemented by minimizing the distance between VGG16 feature maps originated from clean and reconstructed images. As shown in Table C.1, the perceptual loss produces the lowest performance compared to other image-level losses since it is based on feature-level matching and the feature-level matching is not helpful for our purpose, as discussed in Section 3.4. On the other hand, the L1 loss gives the highest performance on both clean and corruption images. Therefore, we choose the L1 loss as a metric of (4) in all experiments.

Loss Types	Architecture	Clean ↑ (%)	Corrupt. ↑ (%)
L1 loss	QualNet50	**82.33**	**72.30**
L2 loss		81.05	71.32
Likelihood loss [28]		81.28	71.37
Adversarial loss [16]		81.26	71.87
Perceptual loss [26]		80.61	70.57

Table C.1. Ablation study on quality losses. We use QualNet50 with ImageNet-200 and evaluate 200-class versions of ImageNet-1K validation set and ImageNet-C [19]. "Clean" and "Corrupt." indicate Top-1 clean accuracy (%) and averaged accuracy on 15 corruption types, respectively. The best results are indicated in bold.

C.2. Effect on λ

We explore the hyperparameter of our method, λ, to confirm whether the performance of our method is sensitive to this value. We conduct experiments with $\lambda = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. Note that our method with $\lambda = 0.0$ is the same as the naïve augmentation scheme. As shown in Fig. C.2, we observe that our method with any λ produces better performance than the naïve augmentation ($\lambda = 0.0$). Furthermore, the high dominance of the quality loss decreases the performance in both clean and corrupted images, particularly on $\lambda = 0.8$ and 1.0.

Methods	λ	Clean ↑ (%)	Corrupt. ↑ (%)
QualNet50	0.0	80.45	70.82
	0.2	82.33	71.96
	0.4	**82.37**	72.17
	0.6	82.33	**72.30**
	0.8	81.83	71.78
	1.0	81.44	71.80

Table C.2. Ablation study on hyperparameter λ. We train QualNet50 with ImageNet-200, varying λ from 0.0 to 1.0 with 0.2 step. We evaluate 200-class versions of ImageNet-1K validation set and ImageNet-C [19]. "Clean" and "Corrupt." indicate Top-1 clean accuracy (%) and averaged accuracy on 15 corruption types, respectively. The best results are indicated in bold.

C.3. Effect on decoder model size: transferability and robustness

We investigate our decoder with respect to the number of layers. For the same accuracy, the invertible classifier requires many layers compared with non-invertible network architecture (e.g., ResNet [17]) due to the invertible property. We use i-RevNet [25] as an invertible network and the number of layers is 300 as default in the paper. We confirm the performance trend when increasing the number of layers in the invertible network architecture (i.e., i-RevNet), as shown in Table C.3. We remark that the performance of mCE, i.e., performance on corrupted images, is improved as the number of layers is increased. Furthermore, the small number of layers is not beneficial to produce performance improvement since it may cause unfavorable guidance to the target classifier. It is experimentally proven that a larger model produces better transferability (i.e., larger decoder increases the transferability) and the transferability is positively correlated with robustness (i.e., larger decoder produces better robustness) [8].

Decoder Types	# Layers	mCE ↓ (%)
None (Naïve Augmentation)	0	52.35
i-RevNet	45	52.86
	78	52.86
	150	52.40
	300	50.60
	450	**49.48**

Table C.3. Ablation study on the network capacity of invertible decoder. We train QualNet50 with different number of layers of the invertible decoder. Different from the other ablation studies, we use ImageNet-1K [5] as a training set and evaluate ImageNet-C [19]. "mCE" shows the performance (%) over 15 corruption types (less is better). The best results are indicated in bold.

C.4. Clean ratio in a mini-batch

The clean ratio in a mini-batch may be important to control the performance between clean and corruption types. Most approaches use a 50% clean ratio in a mini-batch, and we use it for consistency. However, it is worth examining how the clean ratio influences the performance of clean and corruption types. We train our method with the clean ratio varying from 0% to 100% with the step of 25%. As shown in Fig. C.1, when the clean ratio is increased, the clean accuracy (%) is mostly improved, while the corruption accuracy (%) is decreased. Note that the lower clean ratio (e.g., 0% clean ratio) does not guarantee the higher the corruption accuracy while the higher clean ratio (e.g., 75% clean ratio) does not produce higher clean accuracy (see Fig. C.1). Furthermore, we remark that the clean ratio of 50% produces appropriate accuracy (%) on both clean and corrupted images. Hence, the clean ratio of 50% is used for all our experiments.

(a) (b)

Figure C.1. Ablation study on clean ratio. (a) Clean accuracy (%) and (b) Corruption average accuracy (%). We train QualNet50 with ImageNet-200, varying the clean ratio from 0% to 100% with 25% step. We evaluate 200-class versions of ImageNet-1K validation set and ImageNet-C [19].

D. QualNet-LM : alternative for large-margin feature learning

We propose additional variant of our method, called QualNet-LM. The property of the invertibility makes feature learning hard, particularly in the case of requiring more discriminative features. This may be resolved by using better invertible network architecture (if it exists), but we examine more effective way to solve this fundamental issue. In essence, the main idea of our method is to transfer an invertible decoder trained with HQ images. We suggest using the following new two-stage learning scheme for large-margin feature learning: we obtain the decoder f_ψ^{-1} at the first stage, by training the encoder-decoder structure with HQ images as shown in Fig. D.1. Then, the pre-trained decoder f_ψ^{-1} is transferred and frozen for the second stage and we optimize a quality-agnostic classifier f_θ with HQ and LQ images using the frozen decoder f_ψ^{-1}, as shown in Fig. D.1. Note that CosFace [45] is used as a large-margin softmax loss for both stages. Also, we can adopt other large-margin softmax losses such as SphereFace [32], ArcFace [7] and DiscFace [27]. The pre-trained invertible decoder f_ψ^{-1} enables to reconstruct HQ images from the large-margin discriminative features, such that the target classifier f_θ is able to learn the large-margin discriminative features at the second stage. To facilitate preserving clean statistics, the target classifier f_θ for the second stage is initialized by the pre-trained clean encoder at the first stage, and then we freeze the batch normalization [24] parameters of the target classifier f_θ for the second stage training. This scheme not only preserves clean statistics, but also induces the corrupted images into clean statistics. In our experiments, we observe that QualNet-LM produces better performance than the original QualNet in face recognition tasks. Basically, QualNet has trouble in training the decoder with a large angular-margin because the decoder is obtained by training the *invertible classifier* which is directly

influenced on large-margin constraint as shown in Fig. 1. In contrast, QualNet-LM allows for training with a large angular-margin. Namely, the large angular-margin constraint does not directly influence training the decoder in the encoder-decoder structure as in Fig. D.1. Hence, it would be better to use QualNet-LM in face recognition tasks.

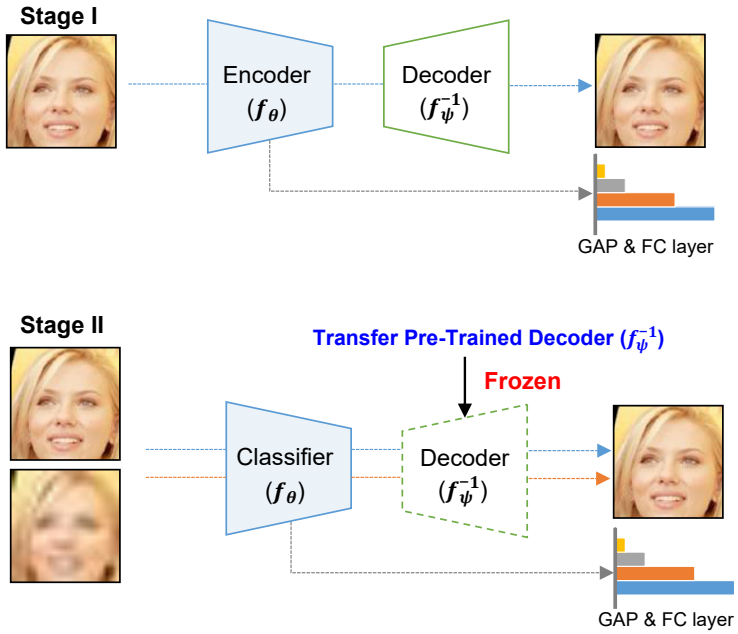

Figure D.1. Quality-agnostic structure for large-margin feature learning (QualNet-LM). Note that we use not only encoder-decoder structure in the first stage but also in the second stage. This structure allows us to train the encoder with a large angular-margin and the decoder that reconstructs only from the large-margin features.