APPENDIX: Leveraging Test-Time Consensus Prediction for Robustness against Unseen Noise

In this appendix, we provide details that could not be included in the main paper owing to space constraints, including: (i) discussion about earlier efforts which are based on different related perspectives; (ii) detailed experiments and results which include architecture details, performance metrics and comparisons of our method with other baseline methods; (iii) detailed ablation studies showing effects of different parts of our proposed method; (iv) visualizations of effect of quantized latent (VQ-VAE) on mitigating noise for sample images from CIFAR10-C, MNIST-C and TinyImageNet-C datasets.

A. Related Work

Robustness against Unseen Noise and Other Corruptions: Over the last 2-3 years, there has been an increased interest to identify and study the issue of the inability of normally trained DNN models when performing on unseen noisy images.Geirhos et al [7] identified that CNNs trained on the standard ImageNet dataset focuses more on textures rather than shape which is fundamentally different from human behavior. They proposed a stylized version of ImageNet and proved that models trained with this dataset learn shape-based representation better, and are more robust against unseen image distortions. Hendrycks et al [10] looked at how models can handle common real-world image corruptions (such as fog, blur, and JPEG compression) as well as noises (Gaussian, shot, impulse, speckle) and propose a comprehensive set of distortions to evaluate real-world robustness. Cubuk et al proposed AutoAugment [2], a reinforcement learning-based method which understands the types of augmentations required for model training using policy gradients, to perform better on corrupted data. Hendrycks et al [12] also proposed AugMix, an image augmentation method derived from AutoAugment, which showed that training a DNN model using diverse set of augmentations improves robustness significantly. These augmentations are constructed by taking a linear combination of different compositional transformations of the original image. Rusak et al [24] proposed a method to generate maximally confusing noise from standard Gaussian noise and train a model to accordingly adapt with the generated noise to work on unseen corruptions. Another recent work [26] showed how covariate shift statistics of the corrupted images can be leveraged, by replacing the training set statistics, to improve robustness substantially. Patch Gaussian [17] is another simple augmentation scheme that adds noise to randomly selected patches in an input image which helps to overcome the trade-off between robustness and accuracy. Though these augmentation methods make models robust against corruptions, it has been recently demonstrated [18] that model robustness for augmentation-based methods drastically falls when test-time distribution is dissimilar to augmented data. In this work, we propose a method to address unseen noise when only training on clean data.

Improving Generalization at Inference: Sun et al [27] recently proposed a training strategy for better generalization during inference time. This method helps the model to adapt to the test distribution by leveraging a self-supervised rotation pretext task, and was shown to produce promising results against unseen corruptions on CIFAR10-C. A more recent paper [1] achieved slightly improved results over test time training [27]. We improve upon these methods, as described in Sec 2, and also compare against them in all our experimental studies. Also note that our method is designed to work for a single data point at test time where data from different unknown distributions are expected to arrive arbitrarily. This makes our work different from batch-based methods such as a recently proposed method [31].

Self-Supervision for Representation Learning: Different self-supervised methods have been proposed to help learn better representations and boost classification accuracy. Tasks such as predicting the relative position of image patches [4], predicting rotation angle [8], recovering color channels [37], solving jigsaw puzzle games [20], and discriminating images created from distortion [5] have been extensively used in recent years. Another class of methods reconstruct images from corrupted versions or just part of it such as denoising autoencoders [30], image inpainting [21], and split-brain autoencoder [38]. Contrastive learning is another paradigm where representations are learned in such a way that similar data points are brought closer and dissimilar data points are pushed further away [32]. Predicting natural ordering or topology of data has also leveraged as pretext tasks in video-based [33, 19, 6], graph based [14, 35], and text-based [22, 3] self-supervised learning. Hendrycks et al [11] were the first to show that self-supervision can be useful for model robustness against adversarial examples, label corruption and common input corruption by helping learn distribution shifts on CIFAR10-C. We build on their success by proposing a test-time strategy that leverages self-supervision and ensembling to address this important challenge.

In addition to the aforementioned perspectives, we use the idea of quantized latents [28] and knowledge distillation (KD) [13] in our extended framework, TTCP++, which we describe in the next section. While KD methods were initially proposed for model compression [23], more recent work [25] have shown the effectiveness of KD in efficient learning with label noise and class imbalance, which we leverage in this work.

B. Experiments and Results

In order to study our method's efficacy on improving model robustness against different kinds of unseen noise,

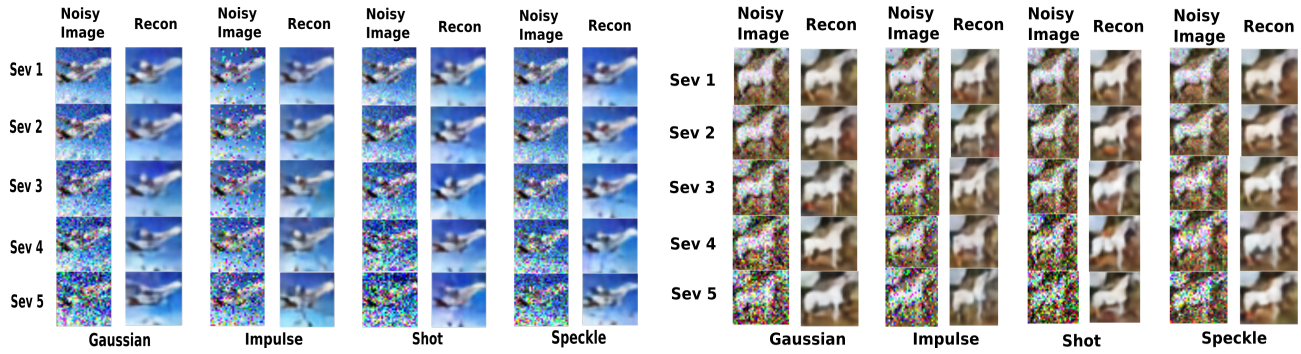


Figure 3. Visualizing effect of quantized latent on mitigating noise for two sample test images from CIFAR10-C dataset (Recon = reconstruction of VQ module). Note that the VQ module can address the noise at all severity levels with no prior knowledge of the noise.

Method	Natural	Baselines JT [11]	TTT [27]	SSDN [1]	OURS TTCP	TTCP++
SL 1	78.30	79.60	80.90	NA	83.18	**83.27**
SL 2	68.30	69.00	71.20	NA	78.44	**81.51**
SL 3	57.80	59.80	62.80	NA	66.88	**79.32**
SL 4	53.60	55.00	58.50	NA	61.81	**77.76**
SL 5	49.50	50.60	54.40	53.00	56.01	**72.21**
Avg	61.50	62.80	65.56	NA	69.26	**78.91**

Table 1. Results with Gaussian noise (SL = Severity Level)

Method	Natural	Baselines JT [11]	TTT [27]	SSDN [1]	OURS TTCP	TTCP++
SL 1	83.00	83.10	83.50	NA	84.71	**84.79**
SL 2	75.70	76.60	77.00	NA	81.86	**82.92**
SL 3	69.30	70.10	71.40	NA	74.29	**81.06**
SL 4	55.20	57.80	60.20	NA	64.09	**77.38**
SL 5	43.90	46.60	50.00	51.00	54.17	**67.14**
Avg	65.42	66.84	68.42	NA	71.82	**78.66**

Table 2. Results with Impulse noise (SL = Severity Level)

Method	Natural	Baselines JT [11]	TTT [27]	SSDN [1]	OURS TTCP	TTCP++
SL 1	82.10	83.40	84.20	NA	85.62	**86.29**
SL 2	77.40	77.40	79.30	NA	82.04	**83.71**
SL 3	64.90	65.60	68.40	NA	72.11	**79.37**
SL 4	60.80	61.70	64.60	NA	67.38	**75.16**
SL 5	52.80	54.70	58.20	56.00	61.03	**66.29**
Avg	67.60	68.56	70.94	NA	73.64	**78.16**

Table 3. Results with Shot noise (SL = Severity Level)

Method	Natural	Baselines JT [11]	TTT [27]	SSDN [1]	OURS TTCP	TTCP++
SL 1	82.85	83.14	83.21	NA	83.67	**84.97**
SL 2	75.67	77.33	79.57	NA	80.05	**83.64**
SL 3	64.19	65.87	68.21	NA	70.02	**80.07**
SL 4	54.86	57.29	59.76	NA	66.41	**78.36**
SL 5	50.78	53.45	55.91	53.00	59.74	**72.61**
Avg	65.67	67.42	69.33	NA	71.98	**79.93**

Table 4. Results with Speckle noise (SL = Severity Level)

Results of different methods with Gaussian, Impulse, Shot and Speckle noises from CIFAR10-C dataset using ResNet-26 architecture

we compare our method's performance with other baseline methods – JT [11], TTT [27] and SSDN [1] – on 4 kinds of noise from CIFAR10-C, Tiny-ImageNet-C and MNIST-C datasets: *Gaussian Noise*, *Shot Noise*, *Impulse Noise* and *Speckle Noise* with 5 increasing levels of severity in each noise category. Our methods, both TTCP and TTCP++, show promising improvement in performance over baseline methods against all these kinds of noise across the datasets. We follow [10] in defining noise as corruptions arising due to Gaussian, shot, impulse and speckle, rather than those caused due to weather and other variations. We however show results of our method on other corruptions too in Sec F.

Architecture Details of Backbone Network: For fair comparison, we used the same backbone architecture used in TTT [27] and SSDN [1] – ResNet-26 [9] – to perform our experiments on CIFAR10-C. For completeness, we also studied the use of WRN 40-2 [36] since JT [11] reported results with WRN 40-2. We use the ResNet-18 backbone network for our studies on ImageNet-C and report results on Severity Level 5 following [27]. We also introduce results for MNIST-C and TinyImageNet-C with LeNet [16] and ResNet-26 [9] backbone architectures respectively. More details of architectures for each dataset are in Sec D.

Architecture Details of Vector Quantization and KD module: We used an encoder network consisting of 2 convolutional layers followed by 2 residual blocks each of which contains 2 convolutional layers. The decoder follows the same architecture in mirrored fashion, and the dimension d of the latent code vector is taken as 512. We used group norm [34] in between convolutional layers in both encoder and decoder modules to handle single sample reconstruction required for TTCP at test-time. WRN 28-10 [36] architecture is used as the teacher network for experiments with CIFAR10-C and Tiny-ImageNet-C datasets, while ResNet-18 and ResNet-152 networks are used as teacher networks for MNIST-C and ImageNet-C datasets respectively. All these pretrained models are available publicly, which we leveraged directly in this work, thereby causing no additional training cost for these models. More details on the architecture is provided in Sec D.

Performance Metrics: Following the benchmark on such noise provided in [10], we report results of our method and baselines on noisy test data for all the mentioned datasets.

Method	Natural	Baselines			OURS	
		JT [11]	TTT [27]	SSDN [1]	TTCP	TTCP++
SL 1	55.61	NA	57.09	NA	58.76	**60.74**
SL 2	45.72	NA	48.97	NA	50.66	**56.38**
SL 3	34.05	NA	39.43	NA	43.92	**49.71**
SL 4	26.81	NA	30.52	NA	33.18	**44.12**
SL 5	21.19	NA	24.91	NA	27.47	**41.23**
Avg	36.67	NA	40.18	NA	42.79	**50.43**

Table 5. Results with Gaussian noise (SL = Severity Level)

Method	Natural	Baselines			OURS	
		JT [11]	TTT [27]	SSDN [1]	TTCP	TTCP++
SL 1	57.87	NA	59.40	NA	59.83	**61.72**
SL 2	48.93	NA	50.23	NA	51.97	**57.27**
SL 3	36.49	NA	41.76	NA	44.78	**50.64**
SL 4	28.34	NA	32.67	NA	36.57	**42.35**
SL 5	24.57	NA	26.54	NA	29.32	**39.18**
Avg	39.24	NA	42.12	NA	44.49	**50.23**

Table 6. Results with Impulse noise (SL = Severity Level)

Method	Natural	Baselines			OURS	
		JT [11]	TTT [27]	SSDN [1]	TTCP	TTCP++
SL 1	58.61	NA	60.41	NA	61.18	**62.98**
SL 2	50.02	NA	52.85	NA	54.69	**58.26**
SL 3	39.15	NA	42.97	NA	45.88	**50.41**
SL 4	30.11	NA	33.81	NA	37.04	**43.68**
SL 5	25.39	NA	27.98	NA	30.16	**39.53**
Avg	40.65	NA	43.60	NA	45.79	**50.97**

Table 7. Results with Shot noise (SL = Severity Level)

Method	Natural	Baselines			OURS	
		JT [11]	TTT [27]	SSDN [1]	TTCP	TTCP++
SL 1	58.24	NA	59.87	NA	60.05	**62.37**
SL 2	48.79	NA	50.39	NA	52.61	**57.84**
SL 3	39.26	NA	41.28	NA	44.99	**50.02**
SL 4	28.03	NA	31.26	NA	36.57	**43.17**
SL 5	23.15	NA	25.29	NA	29.68	**39.21**
Avg	39.49	NA	41.62	NA	44.78	**50.52**

Table 8. Results with Speckle noise (SL = Severity Level)

Results of different methods with Gaussian, Impulse, Shot and Speckle noises on TinyImageNet-C dataset using ResNet-26 architecture

Method	Natural	Baselines			OURS	
		JT [11]	TTT [27]	SSDN [1]	TTCP	TTCP++
Gauss	1.3	2.1	3.1	NA	3.8	**4.5**
Impulse	1.3	2.1	3.5	NA	4.1	**4.7**
Shot	2.0	3.1	4.5	NA	5.2	**5.9**
Clean	68.9	69.1	69.0	NA	**69.7**	68.9

Table 9. Results of different methods with Gaussian, Impulse, Shot noises and Clean images on ImageNet-C dataset using ResNet-18 architecture (Results shown only for SL 5 following [27])

Method	Nat.	JT[11]	TTT[27]	SSDN[1]	TTCP	TTCP++
Clean	91.10	91.90	92.10	90.00	**93.05**	92.01

Table 10. Accuracy on clean CIFAR10 test data using ResNet-26

In case of CIFAR10-C, TinyImageNet-C and MNIST-C, we consider all severity levels (SL) 1 to 5 for Gaussian, Impulse, Shot and Speckle noises as in [10]. For ImageNet-C, we follow [27] and report resutls on SL 5 (most severe).

Results: Tables 1, 2, 3 and 4 report the results of our methods, TTCP and TTCP++, as well as the baseline methods on CIFAR10-C data with Gaussian, impulse, shot and speckle noise respectively. Similar results are shown for Tiny-ImageNet-C [10] in Tables 5, 6, 7 and 8. Results on ImageNet-C with noise of severity level 5 are shown in Table 9. Results on MNIST-C are deferred to Sec E. From these tables, it is evident that our method shows significant improvement in performance over previous methods. Importantly, we note that apart from improved robustness against unseen noise, Table 10 shows that our method does not sacrifice on clean data accuracy too. More results on clean accuracy are included in Sec E. Our experimental findings suggest that TTCP (alone) excelled over TTT [27], showing its capability to be an independent useful module for generalization beyond this work. Our extended method TTCP++ further improves robustness to unseen noise, making it relevant and useful for deployment in in-the-wild settings where unseen noise may be expected.

TTCP vs TTT: Since TTCP and TTT [27] are test-time methods, we performed a comparative study between these two explicitly for better understanding of our contribution. Fig 4 (visualized with sample images) shows results of the methods on CIFAR10-C, where a significant improvement ($\sim$3-4% on average) is noticed for TTCP over TTT. The qualitative results show that as severity level increases, TTCP's improvements are more evident. Similar improvements of TTCP over TTT on MNIST-C, Tiny-ImageNet-C and ImageNet-C datasets are shown in Sec F.

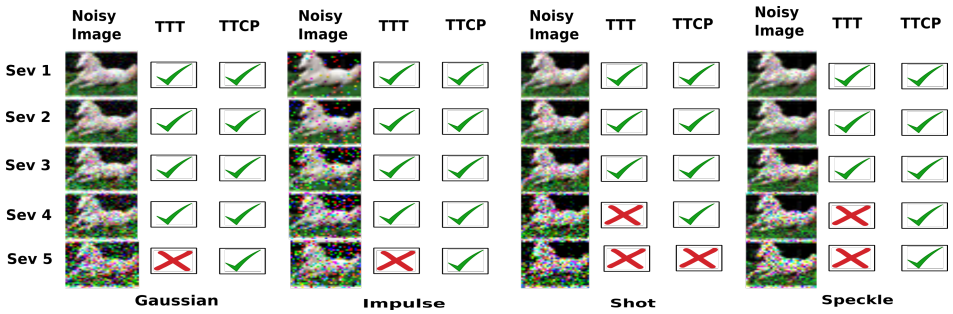
Method	Baselines			OURS
	AA [2]	PG [17]	AM[12]	TTCP++
Gaussian Noise	71.00	81.00	81.00	**82.29**
Impulse Noise	74.00	76.00	**86.00**	81.86
Shot Noise	72.00	74.00	**85.00**	81.62
Speckle Noise	68.00	71.00	78.00	**83.42**
Brown Noise	82.50	69.90	72.20	**82.76**

Table 11. Comparison of our method with augmentation-based methods on different noises on CIFAR10-C using WRN 40-2 architecture

Method	Baselines			OURS
	AA [2]	PG [17]	AM[12]	TTCP++
Gaussian Noise	31.00	33.00	35.00	**36.54**
Impulse Noise	28.00	30.00	33.00	**34.16**
Shot Noise	32.00	32.00	**34.00**	33.02
Speckle Noise	30.00	31.00	35.00	**35.73**
Brown Noise	31.00	29.00	26.00	**34.25**

Table 12. Comparison of our method with augmentation-based methods on different noises on ImageNet-C using ResNet-50 architecture

Computational Overhead: TTCP is real-time, similar to TTT [27], and incurs negligible overhead at test-time. Evident from the results, TTCP by itself provides improvements over baseline state-of-the-art methods. TTCP++ has



Method	TTT	TTCP
Gauss	65.56	**69.26**
Impulse	68.42	**71.82**
Shot	70.94	**73.64**
Speckle	69.33	**71.98**
Clean	92.10	**93.05**

Figure 4. TTT vs TTCP results on improving generalization at inference time. *(Left)* Sample image with different noise at different severity levels; *(Right)* Accuracy on complete test data.

an offline VQ step to obtain the discretized latents which can incur an additional cost. We however found this to be minimal too, considering a simple fully convolutional encoder-decoder architecture (2 conv layers followed by 2 residual blocks with 2 conv layers each) was sufficient for this purpose. On CIFAR10, our entire training (including the offline VQ module) took $\sim$ 1.3x training time over the baselines. (If the VQ step is performed offline for a dataset, this increase goes away too.) Since we use a pre-trained teacher network for the knowledge distillation step, no additional training time overhead was incurred here.

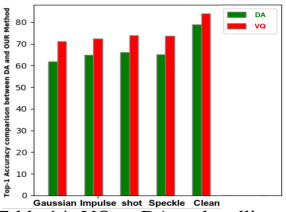
Comparison with Augmentation Methods: We also compare our method against recent augmentation methods i.e. AutoAugment (AA) [2], AugMix (AM) [12] and Patch Gaussian (PG) [17] that have attempted to address similar issues (we do not use [24] and [26] since they did not report results on CIFAR10-C). Following these baselines, we use WRN 40-2 and ResNet-50 architectures for CIFAR10-C and ImageNet-C datasets respectively. The results are reported in Tables 11 and 12. In general, we outperform augmentation methods – especially AA and PG consistently – across all noise categories in both datasets. AM outperforms us in a couple of noise categories, but we note here that our method achieves these results with no training augmentation with models trained only on clean data, making this result significant. Also, as stated in [18], augmentation-based methods perform badly when the distribution of test data differs from the augmented data. Hence, apart from the 4 kinds of noise i.e. Gaussian Noise, Shot Noise, impulse Noise and Speckle Noise, we also experimented with Brown Noise as a less-used noise, and obtain the best result for both CIFAR10-C and ImageNet-C datasets, which shows consistency of our method against any unseen noise. Note again that our method is trained only on clean data.

C. Discussion and Ablation Studies

Effect of different parts of proposed method: Table 13 present the results of our studies of the effect of different parts of our proposed method for Gaussian Noise, Impulse Noise, Shot Noise and Speckle Noise on CIFAR10-C dataset where TTCP++ represents VQ+KD+TTCP (detailed results for each severity level are presented in Sec F). In general, we notice that the idea of using discretized latents using VQ has a significant impact on the performance against unseen noise, as we surmised. Individually, KD helps improve performance, especially when all modules are considered collectively. The last two columns compare between (VQ + KD + TTT) and (VQ + KD + TTCP) which shows the effectiveness of TTCP over TTT when used with other parts of our proposed model. We also observe similar trends for MNIST-C and TinyImageNet-C datasets in Sec

Method	KD	VQ	VQ+KD	VQ+KD+TTT	TTCP++
Gaussian	62.01	71.26	72.30	76.56	**78.91**
Impulse	65.92	72.29	72.83	76.33	**78.66**
Shot	67.86	74.02	74.46	76.27	**78.16**
Speckle	65.94	73.61	74.12	77.20	**79.93**

Table 13. Comparative results (average across 5 severity levels) of diff parts of our method with diff noises on CIFAR10-C using ResNet-26 architecture (detailed results for each SL in Sec F); KD = Standard model training with knowledge distillation. VQ = Accuracy measured with a standard model trained with reconstructed images from vector quantization step. TTCP = Standard model training with our proposed TTCP approach applied during inference. TTCP++ = KD + VQ+ TTCP.



Method	DA	VQ
Gauss	61.74	**71.26**
Impulse	64.95	**72.29**
Shot	66.12	**74.02**
Speckle	65.19	**73.61**
Clean	79.04	**84.91**

Table 14. VQ vs DA on handling noise in CIFAR10-C (results averaged across 5 severity levels)

Noise type	Gauss	Impulse	Shot	Speckle	Clean
Rand. crop	77.57	76.89	76.53	78.06	89.61
Rot.+Rand. crop	78.12	77.54	77.25	78.82	90.28
Rot(OUR)	**78.91**	**78.66**	**78.16**	**79.93**	**92.01**

Table 15. Accuracy comparison of our method with other self-supervision tasks in place of rotation prediction on CIFAR10-C

F.

Effect of Vector Quantization over DA: Denoising autoencoder (DA) was another choice of mitigating noise. We experimented with DA accompanied by natural training and compared this result with the use of VQ module. The results are presented in Table 14 (and adjoining bar plot) for noisy data from CIFAR10-C. VQ performs much better compared to DA for both noisy and clean test data.

Effect of different self-supervised techniques: Rotation prediction pretext task was used as the auxiliary task for all our experiments following earlier related efforts [11][27]. We study the use of other self-supervised tasks such as: (1) random cropping, and (2) random cropping + rotation prediction, keeping everything else same. Our results are shown in Table 15 for CIFAR10-C; the best result was achieved with rotation prediction, which we used in our main results. Exploring other suitable self-supervision tasks will be an important direction of our future work.

D. Architecture Details

We used a VQ architecture similar to what was used for CIFAR10 dataset (Section B in main paper) for our experiments with MNIST and TinyImageNet datasets, shown here. The encoder network consists of 2 convolutional layers followed by 2 residual blocks each of which contains 2 convolutional layers. The decoder follows the same architecture in mirrored fashion. The dimension of the embedding vectors are taken to be 256 and 512 for MNIST and TinyImageNet datasets respectively. We used group norm [34] in between convolutional layers in both encoder and decoder modules to handle single sample reconstruction required for TTCP strategy at inference time.

As explained in Sec 2 of the main paper, a teacher network is introduced to extend the TTCP method to TTCP++. As mentioned in Sec B, we used ResNet-26 as the student (backbone) network for CIFAR10-C (following [27] and [1]), and WRN 28-10 as the teacher network. Going by popular architectures used on MNIST, LeNet and Resnet-18 [9] architectures were used as student and teacher networks respectively for our experiments on the MNIST-C dataset. For experiments on TinyImageNet-C, ResNet-26 and WRN 28-10 architectures were used as student and teacher networks. ResNet-18 and ResNet-152 networks were used as student and teacher networks for the ImageNet-C dataset.

E. Additional Results

MNIST-C: Tables 14, 15, 16 and 17 report comparisons of natural/normal training, test time training [27] and OUR method (comprising all components) on MNIST-C data with all 4 kinds of noise. The results corroborate our claims and show improvement over baseline methods.

CIFAR10-C: In addition to the results on CIFAR10-C in Sec B with ResNet-26 and WRN 28-10 as student and teacher networks respectively, we also studied the performance on CIFAR10C with WRN 40-2 as the student (backbone) architecture (since joint training (JT) [11] uses this architecture). We present results on the 4 kinds of noise: Gaussian, Impulse, Shot and Speckle noise on CIFAR10-C using WRN 40-2 and WRN 28-10 architectures as student and teacher networks respectively in Tables 20, 21, 22 and 23. The values reported for natural training and JT [11] are obtained using batch-normalization [15] (as in their work), whereas results for test time training and our method followed group normalization [34]. We included results of test time training (TTT) [27] on the WRN 40-2 architecture for completeness of analysis (note that [27] reported their results on only ResNet-26 architecture).

Performance on clean data: In continuation to the results presented in Table 10 in the main paper for CIFAR10, we show results on MNIST and TinyImageNet clean data using different components of our method, as well as vanilla test time training (TTT) [27] in Tables 24 and 25 respectively. These results show that our method maintains performance on clean data (in fact, improves performance in certain cases) while providing strong performance on unseen noise.

F. More Ablation Studies

Effect of individual parts of our method for CIFAR10-C, MNIST-C and TinyImageNet-C: In Table 13, we presented results for different components of our framework as an average across the severity levels (due to space constraints). We herein present the results for each severity level in Tables 26, 27, 28 and 29. Similar results are presented in Tables 30,31,32,33 and tables 34,35,36,37 for MNIST-C and TinyImageNet-C datasets respectively. Note that across these results, as the severity level increases, the VQ and KD module add value, demonstrating the need for the TTCP++ framework.

Performance against other corruptions: We show the effect of our methods, TTCP and TTCP++, over TTT against other corruptions in [10] for CIFAR10-C and ImageNet-C in Tables 38 and 39. Note that our methods consistently outperform other baselines on all these corruptions too. In particular, both our methods improve performance over baselines on weather-based corruptions consistently. For corruptions based on camera capture such as zoom and contrast, the VQ-VAE module is not ideal for retrieving the latent data manifold (our TTCP module outperforms baselines in these cases however). Exploring other options to retrieve the true data manifold for such camera capture-based corruptions would be an interesting direction of future work.

TTCP vs TTT on MNIST-C, TinyImageNet-C and ImageNet-C: In addition to the results on CIFAR10-C (presented in the main paper in Sec C), we show significant improvement using TTCP over TTT for MNIST-C,

Method	Natural	Baseline TTT [27]	OURS TTCP	OURS TTCP++
SL 1	90.91	96.72	97.08	**97.18**
SL 2	86.55	92.59	93.70	**94.67**
SL 3	76.05	86.72	88.81	**93.92**
SL 4	70.23	80.11	83.97	**91.11**
SL 5	64.65	75.26	79.21	**89.26**
Avg.	77.67	86.28	88.55	**93.22**

Table 14. Results on Gaussian noise

Method	Natural	Baseline TTT [27]	OURS TTCP	OURS TTCP++
SL 1	89.04	96.59	96.61	**96.65**
SL 2	86.21	91.10	92.54	**95.09**
SL 3	82.23	87.12	90.05	**93.17**
SL 4	76.01	82.17	86.72	**91.46**
SL 5	71.43	78.25	83.28	**90.52**
Avg.	80.98	87.04	89.84	**93.38**

Table 15. Results on Impulse noise

Method	Natural	Baseline TTT [27]	OURS TTCP	OURS TTCP++
SL 1	81.93	96.19	96.23	**96.71**
SL 2	80.43	94.72	95.24	**95.81**
SL 3	77.84	88.34	90.35	**94.57**
SL 4	74.82	81.28	84.70	**93.98**
SL 5	69.71	76.91	79.56	**92.69**
Avg.	76.94	87.48	89.21	**94.75**

Table 16. Results on Shot noise

Method	Natural	Baseline TTT [27]	OURS TTCP	OURS TTCP++
SL 1	87.24	96.34	96.71	**96.79**
SL 2	84.18	92.51	93.78	**94.98**
SL 3	81.87	87.22	89.09	**93.80**
SL 4	76.59	81.14	87.16	**91.89**
SL 5	70.02	75.73	81.59	**91.38**
Avg.	79.98	86.59	89.66	**93.76**

Table 17. Results on Speckle noise

Results of our methods, TTCP and TTCP++, as well as the baseline method, TTT, with Gaussian, Impulse, Shot and Speckle noise on MNIST-C dataset using LeNet and ResNet-18 architectures as student (backbone) and teacher networks respectively. This results show significant improvement of our method over previous method i.e. TTT. TTT = Test time training [27]

TinyImageNet-C and ImageNet-C datasets in Table 18.

Method	MNIST-C TTT [27]	MNIST-C TTCP	Tiny-ImageNet-C TTT [27]	Tiny-ImageNet-C TTCP	ImageNet-C TTT [27]	ImageNet-C TTCP
Gauss	86.28	**88.55**	40.18	**42.79**	3.1	**3.8**
Impulse	87.04	**89.84**	42.12	**44.49**	3.5	**4.1**
Shot	87.48	**89.21**	43.60	**45.79**	4.5	**5.2**
Speckle	86.59	**89.66**	41.62	**44.78**	NA	NA
Clean	99.57	**99.78**	72.11	**72.65**	69.0	**69.7**

Table 18. Comparative effect of TTT and TTCP on improving generalization at inference time for complete noisy test data from MNIST-C, TinyImageNet-C (results averaged across 5 severity levels) and ImageNet-C datasets (results shown for only severity level 5) using LeNet, ResNet-26 and ResNet-18 architectures as backbone networks respectively.

Combining TTCP with AugMix [12]: Considering AugMix [12] performed the best among the augmentation methods (in our results in Sec B), we studied the possibility of combining our method with AugMix by using AugMix during training, and TTCP at inference. Table 19 shows these results on CIFAR10-C and indicates that our method can give further boost in performance when combined with augmentation methods. Exploring this combination further may be another promising direction of future work to obtain models robust to unseen noise.

Visualizing effect of quantized latents:: We explained the importance of quantized latents in mitigating unseen noise in input image with visualizations for CIFAR10-C in Sec 2 in the main paper. Here we present more such visualizations for sample images from MNIST-C (Figure 5) and TinyImageNet-C (Figure 6) to show the effect across datasets. It is observed from the visualizations that quantization helps in removing noise at different severity levels (even with severity level 5), and thus reconstructing noise-free images that can be passed on to the next stage of our pipeline in TTCP++.

Method	TTCP	AugMix [12]	TTCP+AugMix
Gauss	73.84	81.00	**83.29**
Impulse	75.76	86.00	**87.46**
Shot	78.17	85.00	**86.68**
Speckle	77.68	78.00	**80.74**

Table 19. Comparative effect of AugMix [12] and TTCP over both of them individually for complete noisy test data from CIFAR10-C using WRN 40-2 architecture. (results averaged across 5 severity levels.)

Method	Natural	Baselines		OURS	
		JT [11]	TTT [27]	TTCP	TTCP++
SL 1	NA	NA	84.19	86.59	**87.23**
SL 2	NA	NA	74.62	78.92	**84.79**
SL 3	NA	NA	66.09	73.38	**82.91**
SL 4	NA	NA	61.34	68.09	**81.04**
SL 5	NA	NA	55.86	62.25	**75.46**
Avg	42.00	52.00	68.42	73.84	**82.29**

Table 20. Results with Gaussian noise (SL = Severity Level)

Method	Natural	Baselines		OURS	
		JT [11]	TTT [27]	TTCP	TTCP++
SL 1	NA	NA	86.21	87.14	**87.96**
SL 2	NA	NA	80.14	82.61	**85.69**
SL 3	NA	NA	74.38	78.91	**84.57**
SL 4	NA	NA	63.81	70.03	**80.71**
SL 5	NA	NA	53.26	60.11	**70.36**
Avg	55.00	61.00	71.56	75.76	**81.86**

Table 21. Results with Impulse noise (SL = Severity Level)

Method	Natural	Baselines		OURS	
		JT [11]	TTT [27]	TTCP	TTCP++
SL 1	NA	NA	87.35	88.12	**89.97**
SL 2	NA	NA	82.53	84.53	**86.58**
SL 3	NA	NA	71.82	77.68	**83.04**
SL 4	NA	NA	68.14	74.19	**78.31**
SL 5	NA	NA	61.74	66.37	**70.19**
Avg	53.00	63.00	74.32	78.17	**81.62**

Table 22. Results with Shot noise (SL = Severity Level)

Method	Natural	Baselines		OURS	
		JT [11]	TTT [27]	TTCP	TTCP++
SL 1	NA	NA	86.92	87.44	**88.47**
SL 2	NA	NA	81.35	84.26	**86.78**
SL 3	NA	NA	70.95	76.81	**83.86**
SL 4	NA	NA	66.57	73.38	**81.96**
SL 5	NA	NA	59.36	66.51	**76.03**
Avg	59.00	67.00	73.03	77.68	**83.42**

Table 23. Results with Speckle noise (SL = Severity Level)

Results of our methods, TTCP and TTCP++, as well as the baseline methods with Gaussian, Impulse, Shot and Speckle noise on CIFAR10-C dataset using WRN 40-2 and WRN 28-10 architectures as student (backbone) and teacher networks respectively. Only average scores are provided for natural accuracy and JT as reported in [11]. These results show significant improvement of our method over previous methods. JT = Joint Training [11], TTT = Test time training [27]

Method	Nat.	JT[11]	TTT[27]	SSDN[1]	TTCP	TTCP++
Clean	99.03	NA	99.57	NA	**99.78**	99.53

Table 24. Results on clean MNIST test data

Method	Nat.	JT[11]	TTT[27]	SSDN[1]	TTCP	TTCP++
Clean	70.02	NA	72.11	NA	**72.65**	71.73

Table 25. Results on clean TinyImageNet test data

Accuracy comparison on clean MNIST and TinyImageNet test data using LeNet and ResNet-26 architectures respectively. These results show that, apart from improved robustness against unseen noise, our method doesn't sacrifice on clean data accuracy too.

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	78.73	83.18	80.00	81.94	**83.27**
Sev. Level 2	68.71	78.44	77.89	79.12	**81.51**
Sev. Level 3	58.65	66.88	74.49	75.76	**79.32**
Sev. Level 4	54.17	61.81	67.52	68.01	**77.76**
Sev. Level 5	49.81	56.01	56.28	56.67	**72.21**
Avg. Score	62.01	69.26	71.26	72.30	**78.91**

Table 26. Results with Gaussian noise

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	83.01	84.71	80.98	81.14	**84.79**
Sev. Level 2	75.93	81.86	78.25	79.02	**82.92**
Sev. Level 3	69.67	74.29	76.01	76.86	**81.06**
Sev. Level 4	56.01	64.09	68.56	69.21	**77.38**
Sev. Level 5	44.97	54.17	57.69	57.93	**67.14**
Avg. Score	65.92	71.82	72.29	72.83	**78.66**

Table 27. Results with Impulse noise

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	82.43	85.62	81.55	81.76	**86.29**
Sev. Level 2	77.41	82.04	79.69	79.91	**83.71**
Sev. Level 3	65.11	72.11	73.10	73.89	**79.37**
Sev. Level 4	61.02	67.38	71.92	72.45	**75.16**
Sev. Level 5	53.32	61.03	63.95	64.30	**66.29**
Avg. Score	67.86	73.64	74.02	74.46	**78.16**

Table 28. Results with Shot noise

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	83.06	83.67	79.74	80.57	**84.97**
Sev. Level 2	75.98	80.05	79.11	79.70	**83.64**
Sev. Level 3	64.76	70.02	74.32	74.98	**80.07**
Sev. Level 4	55.09	66.41	70.40	70.81	**78.36**
Sev. Level 5	50.82	59.74	64.47	64.54	**72.61**
Avg. Score	65.94	71.98	73.61	74.12	**79.93**

Table 29. Results with Speckle noise

Comparative results of different parts of our method with different noises from CIFAR10-C dataset using ResNet-26 and WRN 28-10 architectures as student(backbone) and teacher networks respectively. KD = Standard model training with knowledge distillation. VQ = Accuracy measured with a standard model trained with reconstructed images from vector quantization step. TTCP = Standard model training with our proposed TTCP approach applied during inference. TTCP++ = KD + VQ + TTCP.

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	90.96	97.08	94.46	94.62	**97.18**
Sev. Level 2	87.21	93.70	93.61	93.98	**94.67**
Sev. Level 3	77.43	88.81	92.72	93.01	**93.92**
Sev. Level 4	70.47	83.97	89.35	89.79	**91.11**
Sev. Level 5	65.06	79.21	86.09	86.89	**89.26**
Avg. Score	78.22	88.55	91.24	91.65	**93.22**

Table 30. Results with Gaussian noise

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	89.38	96.61	95.83	96.03	**96.65**
Sev. Level 2	86.35	92.54	93.87	94.08	**95.09**
Sev. Level 3	82.87	90.05	91.91	92.17	**93.17**
Sev. Level 4	76.29	86.72	87.45	87.82	**91.46**
Sev. Level 5	71.92	83.28	85.24	85.56	**90.52**
Avg. Score	81.36	89.84	90.86	91.13	**93.38**

Table 31. Results with Impulse noise

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	82.03	96.23	94.76	94.85	**96.71**
Sev. Level 2	80.98	95.24	94.04	94.21	**95.81**
Sev. Level 3	78.14	90.35	93.01	93.84	**94.57**
Sev. Level 4	75.41	84.70	92.02	93.07	**93.98**
Sev. Level 5	70.18	79.56	90.68	90.87	**92.69**
Avg. Score	77.34	89.21	92.90	93.36	**94.75**

Table 32. Results with Shot noise

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	87.68	96.71	94.12	94.37	**96.79**
Sev. Level 2	84.21	93.78	93.96	94.18	**94.98**
Sev. Level 3	82.03	89.09	91.89	92.06	**93.80**
Sev. Level 4	76.99	87.16	89.26	89.61	**91.89**
Sev. Level 5	70.86	81.59	86.15	86.52	**91.38**
Avg. Score	80.35	89.66	91.07	91.34	**93.76**

Table 33. Results with Speckle noise

Comparative results of different parts of our method with different noises on MNIST-C dataset using LeNet and ResNet-18 architectures as student(backbone) and teacher networks respectively. KD = Standard model training with knowledge distillation. VQ = Accuracy measured with a standard model trained with reconstructed images from vector quantization step. TTCP = Standard model training with our proposed TTCP approach applied during inference. TTCP++ = KD + VQ + TTCP.

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	56.17	58.76	56.73	56.97	**60.74**
Sev. Level 2	46.34	50.66	47.90	48.39	**56.38**
Sev. Level 3	35.28	43.92	45.61	46.04	**49.71**
Sev. Level 4	26.35	33.18	40.46	40.75	**44.12**
Sev. Level 5	21.53	27.47	35.93	36.29	**41.23**
Avg. Score	37.13	42.79	45.32	45.68	**50.43**

Table 34. Results with Gaussian noise

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	58.03	59.83	58.19	58.35	**61.72**
Sev. Level 2	48.12	51.97	48.62	48.91	**57.27**
Sev. Level 3	37.86	44.78	46.98	47.35	**50.64**
Sev. Level 4	28.41	36.57	40.14	40.78	**42.35**
Sev. Level 5	24.89	29.32	36.68	36.84	**39.18**
Avg. Score	39.46	44.49	46.04	46.45	**50.23**

Table 35. Results with Impulse noise

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	58.92	61.18	59.48	59.67	**62.98**
Sev. Level 2	50.43	54.69	50.74	50.91	**58.26**
Sev. Level 3	39.76	45.88	48.15	48.23	**50.41**
Sev. Level 4	30.54	37.04	40.85	41.06	**43.68**
Sev. Level 5	25.68	30.16	37.52	37.95	**39.53**
Avg. Score	41.06	45.79	47.35	47.56	**50.97**

Table 36. Results with Shot noise

Method	KD	TTCP	VQ	VQ+KD	TTCP++
Sev. Level 1	58.53	60.05	59.16	59.72	**62.37**
Sev. Level 2	48.76	52.61	49.27	49.73	**57.84**
Sev. Level 3	39.47	44.99	47.03	47.61	**50.02**
Sev. Level 4	28.57	36.57	39.92	40.36	**43.17**
Sev. Level 5	23.68	29.68	36.58	36.74	**39.21**
Avg. Score	39.80	44.78	46.39	46.83	**50.52**

Table 37. Results with Speckle noise

Comparative results of different parts of our method with different noises on TinyImageNet-C dataset using ResNet-26 and WRN 28-10 architectures as student(backbone) and teacher networks respectively. KD = Standard model training with knowledge distillation. VQ = Accuracy measured with a standard model trained with reconstructed images from vector quantization step. TTCP = Standard model training with our proposed TTCP approach applied during inference. TTCP++ = KD + VQ + TTCP.

		Baselines		OURS	
Method	Natural	JT [11]	TTT [27]	TTCP	TTCP++
Snow	79.62	80.24	81.04	82.34	**84.67**
Frost	76.32	77.46	78.78	79.83	**81.49**
Fog	85.76	85.84	86.60	87.71	**88.09**
Zoom	81.42	81.72	83.32	**84.65**	81.92
Contrast	86.14	85.78	86.38	**87.43**	86.19

Table 38. Results of CIFAR10-C. (avg. across 5 sev. levels.)

		Baselines		OURS	
Method	Natural	JT [11]	TTT [27]	TTCP	TTCP++
Snow	15.7	15.3	17.1	17.9	**18.7**
Frost	14.9	15.8	17.9	19.2	**20.1**
Fog	15.3	17.0	20.0	21.4	**22.5**
Zoom	16.2	16.0	18.5	**19.4**	16.7
Contrast	9.7	11.0	14.4	**14.9**	11.8

Table 39. Results of ImageNet-C.(only sev. level 5.)

Results of our methods, TTCP and TTCP++, as well as the baseline methods on different corruptions such as weather based (snow, frost, fog) and camera capture based (zoom, contrast) corruptions of CIFAR10-C and ImageNet-C datasets using ResNet-26 and ResNet-18 architectures respectively.

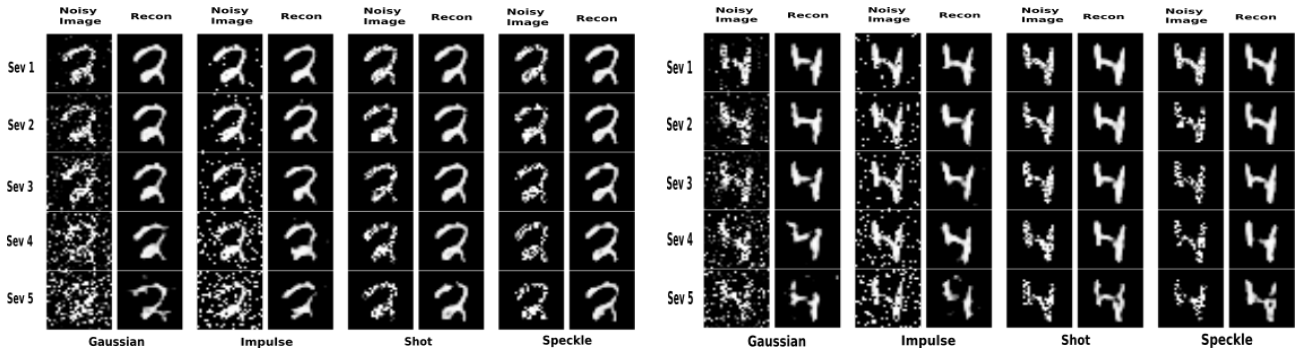


Figure 5. Visualizing effect of quantized latent on mitigating noise for two sample test images from MNIST-C dataset (Recon = reconstruction of VQ module). Note that the VQ module can address the noise at all severity levels with no prior knowledge of the noise.

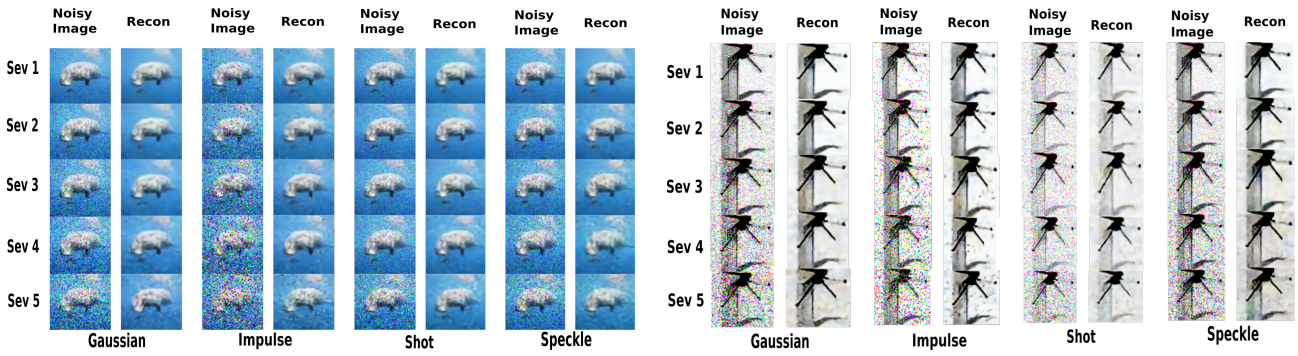


Figure 6. Visualizing effect of quantized latent on mitigating noise for two sample test images from TinyImageNet-C dataset (Recon = reconstruction of VQ module). Note that the VQ module can address the noise at all severity levels with no prior knowledge of the noise.