Leveraging Test-Time Consensus Prediction for Robustness against Unseen Noise

Anindya Sarkar [*], Anirban Sarkar [*], Vineeth N Balasubramanian
Indian Institute of Technology, Hyderabad
anindyasarkar.ece@gmail.com, cs16resch11006@iith.ac.in, vineethnb@iith.ac.in

Abstract

We propose a method to improve DNN robustness against unseen noisy corruptions, such as Gaussian noise, Shot Noise, Impulse Noise, Speckle noise with different levels of severity by leveraging ensemble technique through a consensus based prediction method using self-supervised learning at inference time. We also propose to enhance the model training by considering other aspects of the issue i.e. noise in data and better representation learning which shows even better generalization performance with the consensus based prediction strategy. We report results of each noisy corruption on the standard CIFAR10-C and ImageNet-C benchmark which shows significant boost in performance over previous methods. We also introduce results for MNIST-C and TinyImagenet-C to show usefulness of our method across datasets of different complexities to provide robustness against unseen noise. We show results with different architectures to validate our method against other baseline methods, and also conduct experiments to show the usefulness of each part of our method.

1. Introduction

Generalization performance of Deep Neural Networks (DNNs) is a very important objective, as the networks are susceptible to fail against noise at test time. In recent years, researchers have shown many examples of this kind [10], which raises serious questions about deployment of the seemingly good DNN models in the wild. Although these issues were shown with known types of noise, this problem is actually more challenging because it is difficult to predict what noise will occur at test time.

Recent years have seen a few different efforts in developing models robust to unseen noise. Adversarial joint training [11] was one of the early efforts that focused on improving model robustness by enhancing model generalizability. In particular, this method attempted to find a robust model against noise by adversarial training with a supervised head attached to the model. Though this method performed better than a naturally trained model, there still existed a big gap in performance between test accuracy on clean data and noisy data. More recently, [28] proposed a test-time training method to improve model generalization which showed performance boost over [11] on noisy data. A more recent work [1] further showed improvement over such test-time training [28]. Augmentation-based methods, on the other hand [2, 7, 12, 25] attempt to address model generalization by carefully augmenting the training dataset through different means. However, augmentation methods require a significant increase in the size of the training dataset, and are also known to fail when the test-time distribution differs from the distribution of augmented training data [18]. In this work, we seek to propose a method that can address model robustness to unseen noise by only training on clean data (no additional training data including augmentations).

To this end, we take advantage of ensembled inference through a novel test-time consensus-based prediction method that allows for better generalization at inference. We show that such an approach shows excellent performance against unseen noise, when compared with aforementioned state-of-the-art baselines, especially when no additional data beyond the clean training data is used for training. Building on [28], this method leverages a self-supervision pretext task at test-time to iteratively update the model and predict the class label as a majority vote over multiple predictions of updated models at inference. We call this Test-Time Consensus Prediction (TTCP). In order to further improve model performance against unseen noise, we also propose an extended framework (TTCP++), where a training phase is introduced to: (i) retrieve the latent data manifold from clean data using the idea of quantized latents [29]; and (ii) improve the representations learned by the backbone network used in TTCP via knowledge distillation from a pre-trained teacher network. When quantized latents are used, although the reconstructed images are less noisy, due to the discrete nature of the latent space, the method fails to preserve local texture details during image reconstruction. We leverage knowledge distillation from a pre-trained teacher to help the backbone network learn better representations from such reconstructed images. Our results on multiple benchmark datasets show significant improvement over existing methods, corroborating our claim.

[*]equal contribution

Our key contributions are as follows:

- We propose a novel test-time consensus prediction (TTCP) strategy to achieve better model robustness through improved generalization performance against unseen noise.
- We propose an extended framework, TTCP++, to exploit quantized latents and knowledge distillation in a training phase, to boost the performance of the proposed TTCP method on unseen noise.
- Our results on CIFAR10-C and ImageNet-C are a significant improvement over previous methods based on improved training. We also studied our method on MNIST-C and TinyImagenet-C datasets, which are the first results in this context, and report strong results here too.
- We perform consistently against all kinds of noise on CIFAR10-C and ImageNet-C datasets compared to augmentation-based methods, even without the use of any augmentation and training only on clean data.
- Detailed ablation studies are presented showing the usefulness of each component of our overall TTCP++ framework.

2. Related Work

We now discuss earlier efforts from different related perspectives.

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 [25] 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 [27] 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 [28] 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 [28]. We improve upon these methods, as described in Sec 3, 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-statistics based methods such as a recently proposed method [32]. Another recent paper leveraged test-time ensemble of different transformations to augment the dataset and averaged the output to show performance improvement for adversarial robustness [22]; their objective (adversarial robustness) and method (no model update at test time) is very differnt from our method.

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 [38], 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 [39]. 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 [33]. Predicting natural ordering or topology of data has also leveraged as pretext tasks in video-based [34, 19, 6], graph based [14, 36], and text-based [23, 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,

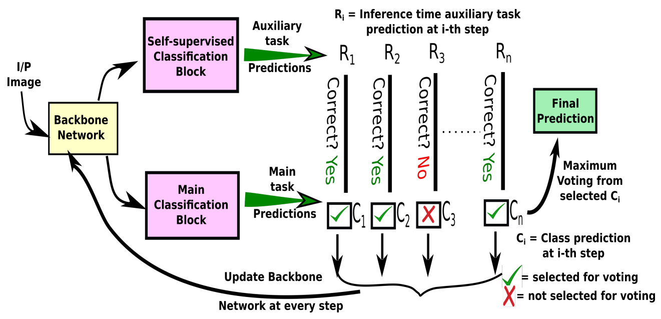

R_i = Inference time auxiliary task prediction at i-th step
I/P Image
Backbone Network
Self-supervised Classification Block
Auxiliary task Predictions
Main Classification Block
Main task Predictions
R1 R2 R3 Rn
Correct? Yes
Correct? Yes
Correct? No
Correct? Yes
C1 C2 C3 Cn
Final Prediction
Maximum Voting from selected C_i
C_i = Class prediction at i-th step
✓ = selected for voting
✗ = not selected for voting
Update Backbone Network at every step

Figure 1. Proposed Test-Time Consensus Prediction (TTCP) method

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 [29] 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 [24], more recent work [26] have shown the effectiveness of KD in efficient learning with label noise and class imbalance, which we leverage in this work.

3. Proposed Methodology

We now introduce our Test-Time Consensus Prediction method (TTCP) towards achieving improved robustness against unseen noise. We then present the extension of our method to TTCP++ which improves the training procedure to further get improvements on noisy datasets. We note that our method involves only training on clean data to achieve the objective of better generalization on corrupted data.

3.1. Test-Time Consensus Prediction (TTCP)

Given training data $\{(x_i, y_i), i = 1, \cdots, n\}$ drawn i.i.d. from a distribution P and model parameters θ, we consider the classification task loss function $\mathcal{L}_m(x_i, y_i, \theta)$ as the main task. We leverage the fact that self-supervised learning empowers model training by improving the intermediate representation with better semantic meaning, which may be beneficial to a downstream task. The labels for a self-supervised task can be generated for free, and a corresponding supervised loss is computed based on the task. We refer to the self-supervised task as *auxiliary task* which yields the loss $\mathcal{L}_{ss}(x_i, y_i, \tilde{\theta})$.

Now, consider a (Y-shaped) DNN represented as a backbone network with two task-specific heads – one for the main classification task and the other for the auxiliary task. Let the model parameters for the backbone network be θ_e, the main task head be θ_m and the auxiliary task head be θ_s, i.e. $\theta = (\theta_e, \theta_m)$ and $\tilde{\theta} = (\theta_e, \theta_s)$. This DNN is trained by minimizing the loss terms for both tasks, l_m and l_{ss}. Assuming the model parameters θ_e and θ_m are already trained using a prior training procedure, following [28], we focus our efforts on the test-time (or inference stage). Given a test data point x forward-propagated through the abovementioned DNN, a gradient step is taken with the objective of minimizing the auxiliary task loss on x i.e.

Algorithm 1: Test-Time Consensus Prediction

```
Input: Test sample $x$, Self-supervision head $f$
       parametrized by $\theta_s$, Classification head $g$
       parametrized by $\theta_m$, Pretrained backbone network
       $e$ parametrized by $\theta_e$, Ground truth auxiliary task
       output in $i^{th}$ step $R_i^*$, Operator $\phi(\cdot)$ which
       converts a softmax output to a one-hot vector,
       Number of classes for main classification task $L$
Output: Predicted class label for test sample $x$
1 Initialize vote counter. f = $f_{pretrained}$ ; e = $e_{pretrained}$;
2 for $i = 1, 2, ..., M$ do
3     $x_{ss}$= $T(x)$; (Random auxiliary transformation)
4     Compute auxiliary task prediction $\hat{R}_i$ = $f(e(x_{ss}))$
5     Compute main classification task prediction $\hat{y}_i$ =
        $g(e(x_{ss}))$
6     Compute classwise vote $V_i = \mathbb{I}[\hat{R}_i = R_i^*]\ \phi(\hat{y}_i)$
7     Update $\theta_e$
8 end
9 Predict class label $\arg\max_{j\in L}(\sum_{i=1}^{M}(V_i))$
```

$$\min_{\theta_e} \mathcal{L}_{ss}(x, \theta_e, \theta_s) \tag{1}$$

For such a gradient step which updates the shared backbone network parameters θ_e^*, the model predicts a class label (main task) through the parameters (θ_e^*, θ_m).

Leverage the capabilities of self-supervision and shared parameter updation at test-time, an auxiliary transformation (e.g. rotation, which defines a corresponding self-supervised task – rotation prediction in this case) is applied to x and passed through the network. At the i^{th} such test-time step, we observe both the prediction of the auxiliary task as well as the main task before the gradient step is taken to update the shared backbone parameters. Let R_i^* denoted the true label for the auxiliary task in the i^{th} step and $\hat{R}_i$ denote the predicted auxiliary task output in the same step. This step is performed a pre-defined number of times, each time with a different auxiliary transformation applied to x under the same self-supervised task (different rotation angle, for example).

At the completion of these steps, we define our final output of test-time consensus prediction as:

$$\arg\max_{j\in L}(\sum_i \mathbb{I}[\hat{R}_i = R_i^*]\ \phi(\hat{y}_i)) \tag{2}$$

where $\phi(.)$ returns a one-hot vector given a softmax output (with the winning position denoted by a 1 and rest zeros), $\mathbb{I}$ is the indicator function, L is the number of classes for the main classification task, $\hat{y}$ is the predicted softmax output of the main classification task, and $C_i = \arg\max_{j \in L} (\hat{y}_i)$. In words, Eqn 2 states that our final output is the consensus of predictions on the main classification output for every transformed input where the auxiliary task head predicts the correct expected output. We term this complete strategy "Test-Time Consensus Prediction (TTCP)". The TTCP method is summarized in Fig 1, and described in Algorithm 1. We note that similar to [28], only a few steps are required at test-time thereby resulting in minimal computational overhead (described further in Sec 4).

3.2. Beyond Test-Time Consensus Prediction (TTCP++)

While TTCP focuses only on test-time, we observe that improvements in obtaining a more robust representation of input data can further help TTCP. To this end, we include a training phase procedure for handling unknown noise at test-time. In particular, we propose a two-staged approach: (i) retrieve the latent data manifold from given clean training data; and (ii) improve the feature representations of the backbone network using knowledge distillation. We now describe each of these steps that together comprise TTCP++.

Retrieving the latent data manifold: Removing noise from data has long been an important topic of research. It is generally hypothesized that a data point x and its noisy version x_{noise} are arbitrarily close on the true data manifold. One of the well-known approaches to retrieve this true data manifold is to use a denoising autoencoder (DA) [31]. Such an approach attempts to remove noise by generating clean images from its noisy version, and works if the type of noise is known beforehand. However, such an approach does not work well (shown in our ablation studies) when handling unknown or unseen noise, given only clean data at training time. We hence instead propose to leverage the idea of quantized latents to mitigate noise in data. Discretization inherently makes two different but nearby points from the given data distribution map to the same point or bring them closer in the latent data manifold. We follow a vector quantization-based approach [29] to achieve this objective.

Our model consists of a standard convolutional encoder-decoder architecture with an intermediate vector quantization (VQ) layer which takes care of building a discrete latent space. More specifically, the encoder network, Z_e, is a fully convolutional neural network which maps input images to an output feature map of size $w \times h \times d$. This provides $w \times h$ d-dimensional vectors, each of which is mapped to one latent code from a set of k learned latent codes $\{e_1, \cdots, e_k\} \in \mathbb{R}^d$ through a mapping (VQ) layer. This is achieved by minimizing the L_2-distance between each of the d-dimensional vectors (obtained as output of Z_e) with the latent codes e_is, i.e.

$$\hat{Z}_e(p,q,:) = e_j, \text{ where } j = \arg\min_{i \in \{1,\cdots,k\}} ||Z_e(p,q,:) - e_i||_2 \quad (3)$$

where Z_e denotes $Z_e(x)$ for a given input x, p and q are indices over $w \times h$ d-dimensional vectors. The output of the encoder hence is $\hat{Z}_e(x)$, a $w \times h \times d$ feature map, where the depth vector at each pixel is replaced by the nearest latent code vector. $\hat{Z}_e(x)$ is then provided as input to a fully convolutional decoder, Z_d, which attempts to reconstruct the input image.

Due to the discrete bottleneck layer in between, the training of this model is not straightforward. The training objective includes the reconstruction loss (mean squared error) and two VQ-based loss terms which guarantee that encoder outputs stay close to the embedding vector entries they are matched to, as below:

$$\arg\min_{\hat{Z}_e, Z_d, \{e_i\}} \log p\big(\tilde{x}|\hat{Z}_e(x)\big) + ||Z_e(x) - \overline{\hat{Z}_e(x)}||^2 + ||\overline{Z_e(x)} - \hat{Z}_e(x)||^2 \quad (4)$$

where $\overline{\cdot}$ denotes the stop gradient operation. A vector under stop gradient operation corresponds to normal forward pass, but no updation during backpropagation. We follow [29] for the rest of the training procedure.

Importantly, we use clean training data to learn vector quantized latents and show that the trained model works well with unseen noise at test-time. We find that this model is capable of removing local textures (including noise), but keeps the global content of an image while reconstructing the image. (We note that this step is offline, and can be done prior to the training of the backbone network used in TTCP.) We present sample visualizations in Fig. 3 for different noises on images from CIFAR10-C dataset to show the potential of this method. More such visualizations from Tiny-ImageNet-C and MNIST-C datasets are presented in the Appendix.

Learning robust representations: While the above step captures global details of the original image, it also misses texture details of original image (as shown in Fig 3). Due to this lossy nature of the reconstructed images, we noticed that the models, trained with these images – while handling unseen noise – report a drop in clean test accuracy (∼9–10%) compared to a model trained on normal data.

To address this issue, we consider a pretrained (on clean data) "Teacher Network". Our objective herein is to leverage the guidance of the teacher network to help the student model in learning refined representations from images only with the global content obtained as output of the quantized latent (VQ) step above. The said purpose is achieved by incorporating the following objective:

$$\mathcal{L}_{KD} = ||\text{logit}(T(x)) - \text{logit}(S(x_{recons}))||_2 \quad (5)$$

where $T(\cdot)$ denotes the pretrained teacher network, $S(\cdot)$ denotes our backbone network from TTCP (student network

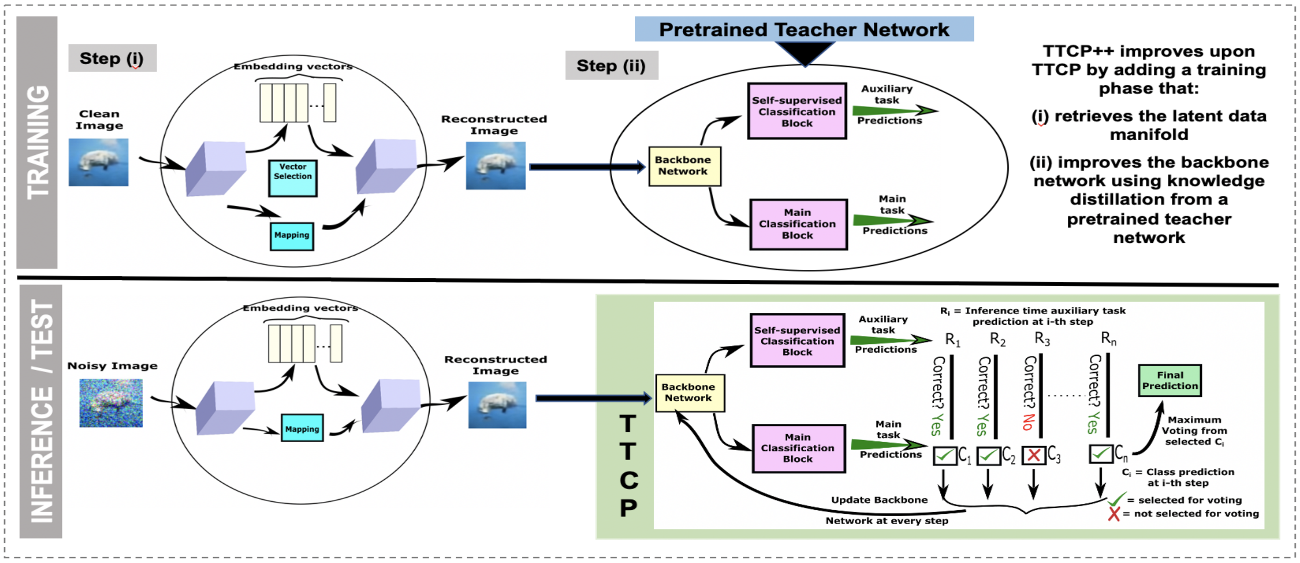

TRAINING
Step (i)
Clean Image
Embedding vectors
Vector Selection
Mapping
Reconstructed Image
Step (ii)
Pretrained Teacher Network
Backbone Network
Self-supervised Classification Block
Auxiliary task Predictions
Main Classification Block
Main task Predictions
TTCP++ improves upon TTCP by adding a training phase that:
(i) retrieves the latent data manifold
(ii) improves the backbone network using knowledge distillation from a pretrained teacher network
INFERENCE / TEST
Noisy Image
TTCP
R_i = Inference time auxiliary task prediction at i-th step
Correct? Yes
Correct? No
Final Prediction
Maximum Voting from selected C_i
C_i = Class prediction at i-th step
Update Backbone Network at every step
✓ = selected for voting
✗ = not selected for voting

Figure 2. Overall framework of TTCP++

in this context), logit$(\cdot)$ represents the logits of the corresponding network, and x_{recons} denotes the reconstructed image obtained from the previous step when x is provided as input. The student network architecture here follows a Y-shape structure (similar to TTCP, for later use of TTCP at test-time) and contains a *self-supervision head* and a *classification head* for auxiliary task prediction and classification respectively. Both these heads follow a shared backbone network. Altogether, the student network is trained with standard classification loss (cross-entropy loss), self-supervision loss (auxiliary prediction loss) and logit similarity loss (minimizing the L_2 distance of logits generated by teacher and student), as given below:

$$\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{SS} + \mathcal{L}_{KD} \tag{6}$$

where $\mathcal{L}_{CE}$ denotes cross entropy loss and $\mathcal{L}_{SS}$ denotes self supervision loss, as before in Sec 3.1.

Inference/Test-time: After the training phase, during inference, we input the test data through the VQ module to obtain the reconstructed image. This is then input to the TTCP module to obtain the final prediction as in Sec 3.1. We name this extension TTCP++ when we consider vector quantization and a pretrained teacher network for model training along with TTCP during inference time. Adding this training phase helps improve clean accuracy by $\sim$3%. Additionally, with this training phase, TTCP during inference phase also achieves improvements on clean test accuracy ($\sim$3.5%) as well as improvements on unseen noisy test data (ranging from 6-10% across different noise). Our results are discussed in detail in Sec 4, and ablation studies of individual components and other combinations are studied in Sec 5.

4. Experiments and Results

We conducted a comprehensive suite of experiments and ablation studies, which we report in this section and in Sec 5. We report results with our method on CIFAR10-C, Tiny-ImageNet-C, ImageNet-C and MNIST-C datasets [10], going beyond earlier related methods – joint training (JT) [11], test-time training (TTT) [28] and SSDN [1] – which focused on CIFAR10-C. Our results with Tiny-Imagenet-C and MNIST-C datasets are the first effort in this direction to the best of our knowledge. In order to study our method's efficacy on improving model robustness against different kinds of unseen noise, we compare our method's performance with other baseline methods – JT [11], TTT [28] 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 the Appendix.

Architecture Details of Backbone Network: For fair comparison, we used the same backbone architecture used in TTT [28] and SSDN [1] – ResNet-26 [9] – to perform our experiments on CIFAR10-C. For completeness, we also studied the use of WRN 40-2 [37] 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 [28]. We also introduce results for MNIST-C and TinyImageNet-C with LeNet [16] and ResNet-26 [9] backbone architectures respectively. More

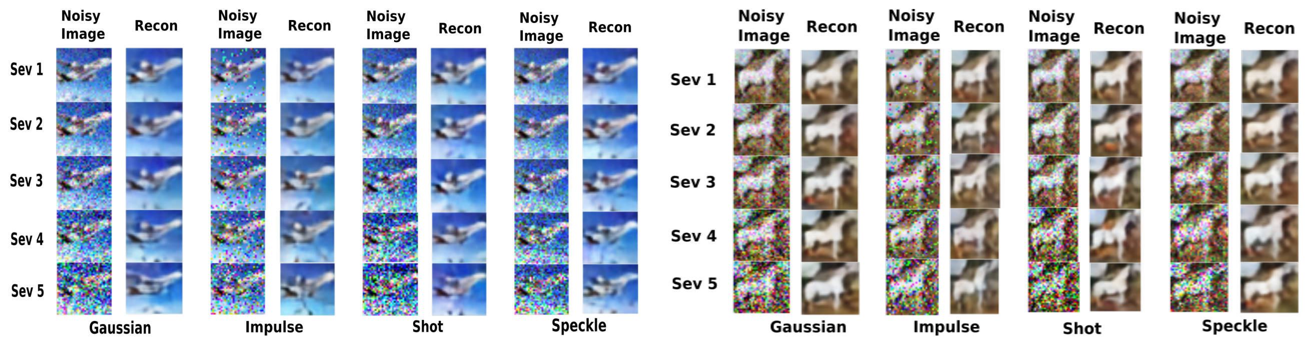


Figure 3. Visualizing effect of quantized latent on mitigating noise for two sample test images from CIFAR10-C dataset (Recon = reconstruction of VQ module). Note that the VQ module can address the noise at all severity levels with no prior knowledge of the noise.

Method	Natural	Baselines			OURS	
		JT [11]	TTT [28]	SSDN [1]	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			OURS	
		JT [11]	TTT [28]	SSDN [1]	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			OURS	
		JT [11]	TTT [28]	SSDN [1]	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			OURS	
		JT [11]	TTT [28]	SSDN [1]	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)

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

details of architectures for each dataset are in the Appendix.

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 [35] in between convolutional layers in both encoder and decoder modules to handle single sample reconstruction required for TTCP at test-time. WRN 28-10 [37] 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 the Appendix.

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. 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 [28] 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 the Appendix owing to space constraints. 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 the Appendix. Our experimental findings suggest that TTCP (alone) excelled over TTT [28], 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 [28] are test-time

Method	Natural	Baselines JT [11]	TTT [28]	SSDN [1]	OURS 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 JT [11]	TTT [28]	SSDN [1]	OURS 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)

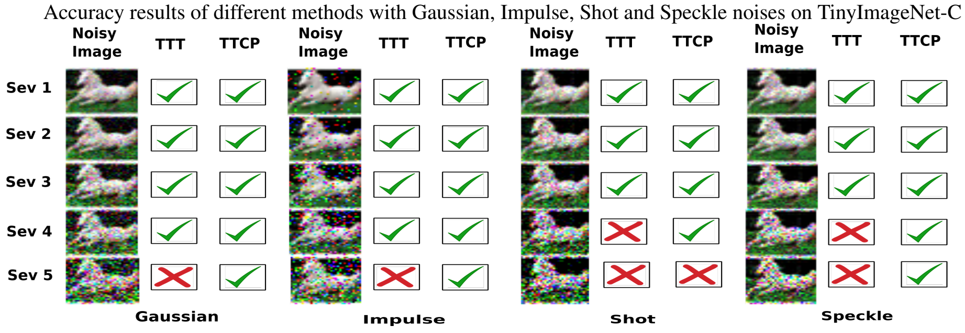
Method	Natural	Baselines JT [11]	TTT [28]	SSDN [1]	OURS TTCP	TTCP++
SL 1	58.61	NA	60.41	NA	61.18	**62.98**
SL 2	50.02	NA	52.85	NA	54.69	**58.26**
SL 3	39.15	NA	42.97	NA	45.88	**50.41**
SL 4	30.11	NA	33.81	NA	37.04	**43.68**
SL 5	25.39	NA	27.98	NA	30.16	**39.53**
Avg	40.65	NA	43.60	NA	45.79	**50.97**

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

Method	Natural	Baselines JT [11]	TTT [28]	SSDN [1]	OURS 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)

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



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.

Method	Natural	Baselines JT [11]	TTT [28]	SSDN [1]	OURS 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. Accuracy results of different methods with Gaussian, Impulse, Shot noises and Clean images on ImageNet-C dataset using ResNet-18 architecture (Results shown for SL 5 following [28])

Method	Nat.	JT[11]	TTT[28]	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

Method	Baselines AA [2]	PG [17]	AM[12]	OURS 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. Accuracy comparison with augmentation-based methods on different noises on CIFAR10-C using WRN 40-2 architecture

Method	Baselines AA [2]	PG [17]	AM[12]	OURS 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. Accuracy comparison with augmentation-based methods on different noises on ImageNet-C using ResNet-50 architecture

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 (∼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 the Appendix.

Computational Overhead: TTCP is real-time, similar to TTT [28], 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 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.

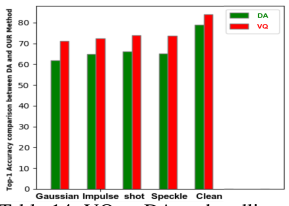
Comparison with Augmentation Methods: We also compare our method against recent augmentation methods i.e. AutoAugment (AA) [2], AugMix (AM) [12] and Patch Gaussian (PG) [17] that have attempted to address similar issues (we do not use [25] and [27] since they did not report results on CIFAR10-C). Following 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.

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 (accuracy averaged 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 Appendix); KD = Std model training with knowledge distillation. VQ = Std model trained with reconstructed images from vector quantization step. TTCP = Std model training with proposed TTCP approach applied at inference. TTCP++ = KD + VQ+ TTCP.

5. Discussion and Ablation Studies

Effect of different parts of proposed method: Table 13 presents results of our studies on the effect of different parts



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 (accuracy 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

of our method for Gaussian, Impulse, Shot and Speckle Noise on CIFAR10-C dataset where TTCP++ represents VQ+KD+TTCP (detailed results for each severity level are presented in the Appendix). 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), showing 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 the Appendix.

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][28]. 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.

6. Conclusions and Future Work

Most real-world environments are inherently noisy, thus hindering DNN models from being deployed in real-world applications, especially in in-the-wild with unknown or unseen noise. In this work, we propose a simple yet effective test-time consensus prediction (TTCP) approach that addresses model robustness to robust noise with training only on clean data. We further propose an extended version, TTCP++, where we add a training phase based on quantized latents and knowledge distillation, that helps improve the performance of TTCP further on unseen noise. Our comprehensive results on multiple benchmark datasets against SOTA baselines show significant promise for deploying DNN models in in-the-wild with unseen noise.

Acknowledgements. This work has been partly supported by the Govt of India UAY program, Honeywell and a Google Research Scholar Award.

References

[1] Tomer Cohen, Noy Shulman, Hai Morgenstern, Roey Mechrez, and Erez Farhan. Self-supervised dynamic networks for covariate shift robustness. *arXiv preprint arXiv:2006.03952*, 2020.

[2] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *Proceed-ings of IEEE Conference on Computer Vision and PatternRecognition*, 2018.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[5] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747, 2015.

[6] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3636–3645, 2017.

[7] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *International Conference on Learning Representations*, 2018.

[8] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[10] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[11] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems*, pages 15663–15674, 2019.

[12] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *Advances in Neural Information Processing Systems*, 2019.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[14] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin D Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv preprint arXiv:1906.02611*, 2019.

[18] Eric Mintun, Alexander Kirillov, and Saining Xie. On interaction between augmentations and corruptions in natural corruption robustness. *arXiv preprint arXiv:2102.11273*, 2021.

[19] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[20] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[21] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[22] Juan C Pérez, Motasem Alfarra, Guillaume Jeanneret, Laura Rueda, Ali Thabet, Bernard Ghanem, and Pablo Arbeláez. Enhancing adversarial robustness via test-time transformation ensembling. *arXiv preprint arXiv:2107.14110*, 2021.

[23] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[24] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[25] Evgenia Rusak, Lukas Schott, Roland S Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. A simple way to make neural networks robust against diverse image corruptions. *European Conference on Computer Vision*, 2020.

[26] Fahad Sarfraz, Elahe Arani, and Bahram Zonooz. Knowledge distillation beyond model compression. *arXiv preprint arXiv:2007.01922*, 2020.

[27] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33, 2020.

[28] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, 2020.

[29] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

[30] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[31] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, page 1096–1103, 2008.

[32] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

[33] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2015.

[34] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018.

[35] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[36] Han Yang, Xiao Yan, Xinyan Dai, and James Cheng. Self-enhanced gnn: Improving graph neural networks using model outputs. *arXiv preprint arXiv:2002.07518*, 2020.

[37] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[38] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[39] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017.