Appendix

A Broader Impact

As previously mentioned, the goal of our work is to create an image classification system which is robust to various distortions on the input data, which has potential applications on improving the practical use of neural network architectures. However, we have to point out some ethical considerations for the use of our work. On the one hand, allowing more applications to access the power of deep learning may not necessarily have a positive impact, since this largely depends on the goals of the application itself. On the other hand, care should be taken when choosing the dataset on which to train the student encoder, as well as the model to serve as the teacher encoder. Since our proposed process aims to distill information from the teacher, as well as use a specific dataset to perform classification, the presence of biases in either may carry over to our student encoder, which is undesirable. It is paramount for the ethical application of our work to distill information from models without any such biases.

B Proof of Proposition 1

Proposition 2. *If R is a minimizer of the uniformity term*

$$R \in \arg\min_{f} \sum_{i=1}^{N} \log \sum_{j=1}^{N} \exp \frac{\langle f(x_i), f(x_j) \rangle}{\tau},$$

then any encoder $S^* \in \arg\min_S \hat{\mathcal{L}}^{\mathsf{contr}}(S; \tau, R, A)$ exactly recovers the target embedding, $S^*(A(x_i)) = R(x_i)$ for all x_i in the training set.

Proof. We are interested in the minimizers of the function $\hat{\mathcal{L}}^{\mathsf{contr}}$. To simplify notation, we denote $R(x_i)$ as R_i, $S(A(x_i))$ as S_i and

$$H_R(x) := \sum_j \exp \langle x, R_j \rangle / \tau.$$

We also denote

$$F_i(S_i) := -\langle S_i, R_i \rangle + \tau \log H_R(S_i)$$

such that

$$\hat{\mathcal{L}}^{\mathsf{contr}}(S; \tau, R, A) = \frac{1}{\tau \cdot N} \sum_{j=1}^{N} F_i(S_i).$$

Since each S_i only appears in one term of the above sum, it suffices to show that F_i is uniquely minimized by the argument R_i. The only assumption we make on R_i follows from [42] which shows that if R minimizes the uniformity term, then $\sum_j R_j = 0$.

We prove the claim directly by showing that $F_i(S_i) - F_i(R_i) > 0$ for any S_i with $\|S_i\| = 1$ and $S_i \neq R_i$. Fix some S_i with norm 1 and suppose that $\langle S_i, R_i \rangle = 1 - \delta$. Then from an argument by cosine distance, we see that replacing R_i with S_i cannot alter the dot product $\langle R_i, R_j \rangle$ by more than δ for any j: $|\langle R_i - S_i, R_j \rangle| \leq \delta$ for all R_j. Using optimality of R, we claim there exists some j such that this is strict on one side. There must exist a j such that

$$\langle R_j, S_i \rangle > \langle R_j, R_i \rangle - \delta,$$

because if not, then $\langle R_j, S_i \rangle = \langle R_j, R_i \rangle - \delta$ for all j. Summing both sides over j and using the fact that $\sum_j R_j = 0$ generates the contradiction that $0 = -N\delta$.

Now we decompose the $H_R(S_i)$ term inside $F_i(S_i)$:

$$\begin{aligned} H_R(S_i) &= \sum_j \exp\left(\langle S_i, R_j \rangle / \tau\right) \\ &> e^{-\delta/\tau} \cdot \sum_j \exp\left(\langle R_i, R_j \rangle / \tau\right) \\ &= e^{-\delta/\tau} \cdot H_R(R_i), \end{aligned}$$

where the inequality follows from $\exp\left(\langle S_i, R_j\rangle/\tau\right) \geq \exp\left(\langle R_i, R_j\rangle/\tau - \delta/\tau\right)$ for all j, where it is strict for at least one j. We can now directly compare $F_i(S_i)$ to $F_i(R_i)$ as

$$\begin{aligned} F_i(x) - F_i(R_i) &= \langle R_i, R_i - S_i\rangle + \tau \cdot (\log H_R(S_i) - \log H_R(R_i)) \\ &= \delta + \tau \cdot (\log H_R(S_i) - \log H_R(R_i)) \\ &> \delta + \tau \cdot \log e^{-\delta/\tau} = 0. \end{aligned}$$

Hence, f_i has a unique global minimizer at R_i. □

C Variants on the Uniformity Term

As mentioned in the main text, in Section 3, we consider variations on the uniformity term in $\hat{\mathcal{L}}^{\mathsf{contr}}$. We explicitly define these variations here. Recall that

$$\hat{\mathcal{L}}^{\mathsf{contr}}(S; \tau, R, A) := \frac{1}{\tau}\hat{\mathcal{L}}^{\mathsf{MSE}}(S; R, A) + \hat{\mathcal{L}}^{\mathsf{unif}}(S; \tau, R, A).$$

In the main text, we use the 'student vs. teacher' uniformity loss. We explicitly define this variant and all others considered in the following list. We abuse notation slightly and use the function $K_\tau(\cdot, \cdot)$ defined as:

$$K_\tau(y, z) := \exp\left(\langle y, z\rangle/\tau\right),$$

noting that in the main text the arguments to K were *indices* and here they the *embedding vectors* themselves. As above, we let $S(A(x_i))$ and $R(x_i)$ be denoted by S_i, R_i respectively.

- **Student vs. Teacher:** This loss compares the noisy student embeddings to the clean teacher embeddings, denoted as

$$\hat{\mathcal{L}}^{\mathsf{unif}}_{ST}(S; \tau, R, A) := \frac{1}{N}\sum_i \log \sum_j K_\tau(S_i, R_j)$$

- **Student vs. Student:** This loss compares pairs of noisy student embeddings:

$$\hat{\mathcal{L}}^{\mathsf{unif}}_{ST}(S; \tau, R, A) := \frac{1}{N}\sum_i \log \sum_{j \neq i} K_\tau(S_i, S_j)$$

- **Student vs. Both:** This loss combines the above two losses, where the combination occurs *inside the logarithm*:

$$\hat{\mathcal{L}}^{\mathsf{unif}}_{SB}(S; \tau, R, A) := \frac{1}{N}\sum_i \log \left(\sum_{j \neq i} K_\tau(S_i, S_j) + \sum_j K_\tau(S_i, R_j)\right)$$

- **NT-XEnt:** This loss includes terms for the pairwise similarity between all $4N^2 - 2N$ pairs of student and teacher embeddings. We denote this as $NT - XEnt$, because in the contrastive setting where data is provided in terms of batches of paired positive examples $\{x_i, x_i^+\}$, all pairs are considered in the contrastive loss.

$$\begin{aligned} \hat{\mathcal{L}}^{\mathsf{unif}}_{NT}(S; \tau, R, A) := \frac{1}{N}\sum_i \Bigg(&\log \left(\sum_{j \neq i} K_\tau(S_i, S_j) + \sum_j K_\tau(S_i, R_j)\right) + \\ &\log \left(\sum_j K_\tau(R_i, S_j) + \sum_{j \neq i} K_\tau(R_i, R_j)\right)\Bigg). \end{aligned}$$

D Further Experiments and Experimental Details

D.1 Training Details

Datasets For all experiments, we pre-train the robust encoder as well as the baseline using a randomly chosen 100-class subset of the ImageNet dataset [36]. ImageNet consists of 1,000 classes of objects, with 1.2M training images and 50K validation images. The portion we use is the same subset of ImageNet used by [39], and contains 126,689 training and 5,000 validation images. We refer to this dataset as ImageNet-100.

For the transfer learning task, we make use of five datasets: (1) CIFAR-10 [28], (2) CIFAR-100 [28], and (3) STL-10 [8]. (4) The COVID-19 Chest X-ray dataset [9], which consists of X-rays taken of patients with healthy lungs as well as lungs affected by pneumonia resulting from COVID-19. The dataset contains 5,286 training images and 624 test images. (5) We generate another random 100-class subset of ImageNet [36] from the remaining 900 classes we did not use for ImageNet-100, which we refer to as ImageNet-100B. This subset contains 128,987 training images and 5,000 validation images.

Image Pre-Processing During contrastive training, we randomly crop the image and resize it to a height and width of 224 pixels, then apply a random horizontal flip. The resultant image is normalized to have a mean of 0 and standard deviation 1 in each color channel before being fed to the input of the teacher. A copy of the randomly-cropped and flipped image is distorted with a given forward operator and normalized to feed to the student input. We pre-process the images for training the baseline by applying a random crop, resizing to 224×224 pixels, applying a given distortion, and normalizing the pixel values. During validation and testing, we resize each image to a height and width of 256 pixels, take a center crop of 224×224 pixels, apply a distortion, and finally normalize the image. We vary the train and test distortions depending on the setting we evaluate.

For images from CIFAR-10, CIFAR-100, and STL-10, we pre-process the training images by resizing to 224×224 pixels, applying a random horizontal flip, distorting with a given forward operator, and finally normalizing the images. For validation images from these datasets we perform the same process without the random flip. For training images from the COVID-19 X-ray dataset and ImageNet-100B, we apply a random crop, resize to 224×224 pixels, apply a given distortion, and normalize the pixel values. For validation images from these datasets, instead of a random crop and resize, we resize to 256×256 pixels and take a 224×224 center crop of the image, then apply a distortion and normalize.

We apply random pixel masking, Gaussian blur, and additive Gaussian noise as our distortions in the various settings we evaluate. Random pixel masking sets the intensity of a randomly-chosen set of pixels in an image to 0. Gaussian blur convolves the image with an isotropic Gaussian kernel, and additive Gaussian noise applies additive white gaussian noise to each each pixel in the image.

Training Hyperparameters We train our method and the supervised baseline using the Adam optimizer with default values for β and a cosine learning rate schedule [26]. For the baseline, we use a learning rate of 0.001 and a batch size of 64. For our method, we use a learning rate of 0.0003, weight decay of 0.0001 and a batch size of 256, and we set the temperature τ from Eq (3) to be 0.1. We train both the baseline and our method for 25 epochs. The baseline is optimized using cross entropy loss.

We train a linear classifier on top of the learned representations for our method and the baseline for each set of experiments. The linear classification layer is optimized with the Adam optimizer, using a learning rate of 0.001 and a batch size of 128, and is trained for 10 epochs. For label-efficiency experiments, we lower the batch size to 8 to compensate for the decrease in the data used. We freeze the learned backbone network during training of the classifier. The classifier is trained using images that have the same distortions as were used to train the backbone network. During transfer learning using the baseline, we remove the classification layer used on ImageNet-100 pre-training and replace it with a randomly-initialized layer with appropriate dimension.

We choose the hyperparameters for each model using a linear search over several values for each hyperparameter, based on the highest mean validation accuracy we achieve on ImageNet-100 after one epoch of training on each distortion type. Due to computational limitations, we did not use a separate validation set for hyperparameter tuning, but rather extrapolated from these results over a single epoch of training. The values searched over are as follows:

- The learning rate for our method was searched in the range of $[10^{-4}, 10^{-2}]$. For the baseline, the search was over $[10^{-5}, 5 \cdot 10^{-1}]$.
- The weight decay was searched in the range $[10^{-4}, 10^{-3}]$.

- The temperature parameter τ was searched in the range $[0.1, 1]$.

The batch size was set as high as possible with our computing hardware. For our method, this was done since this form of contrastive loss benefits from greater batch size. For the baseline, this choice improved performance. All experiments were performed on a system with 4 Nvidia Quadro RTX5000 GPUs, 2 Intel Xeon E5-2620 v4 CPUs, and 128GB of RAM. Experiments using CLIP networks were run using 16-bit floating point models and data. We use the PyTorch implementation of the supervised ImageNet-trained ResNet-101 [34].

D.2 Top-5 Results for Main Experiments

For sake of completeness, we provide the full metrics for our experiments, which also include top-5 accuracies for the experiments performed. The conclusions we can draw from the top-1 accuracy results do not change when we look at the top-5 accuracies.

For the transfer learning task on COVID X-ray data, we present the area under the receiver operating characteristic curve (AUC). This dataset presents a binary classification problem, so top-5 metrics do not apply.

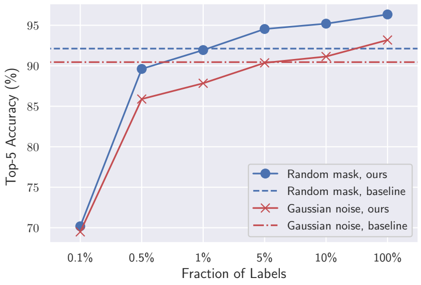

Figure 7: **Top-5 accuracies using a varying fraction of labeled samples to train a linear probe.** We train robust encoders on images with 90% random pixel masking and additive Gaussian noise with standard deviation 0.5, and fit a linear classifier on the learned representations using varying fractions of labeled training samples. We compare to a supervised baseline that uses all of the labeled training samples. Results are averaged over 10 random instantiations of corruptions on the ImageNet-100 validation dataset. We omit error bars as standard error is insignificant.

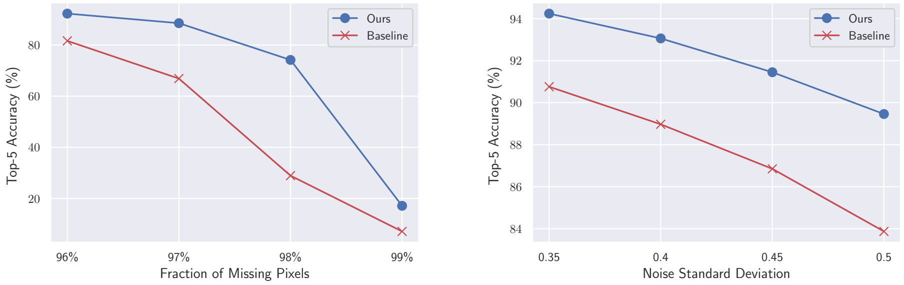

Figure 8: **Top-5 accuracies for images with varying corruption levels using models trained on a range of levels.** In the left figure, we compare our robust model with a baseline, both trained on images with 50% to 95% random pixel masking. In the right figure, each model is trained on images with additive Gaussian noise with random standard deviation from 0.1 to 0.3. We evaluate the models on images with more severe corruptions than applied during training. Results are averaged over 10 random instantiations of corruptions on the ImageNet-100 validation dataset. We omit error bars as standard error is insignificant.

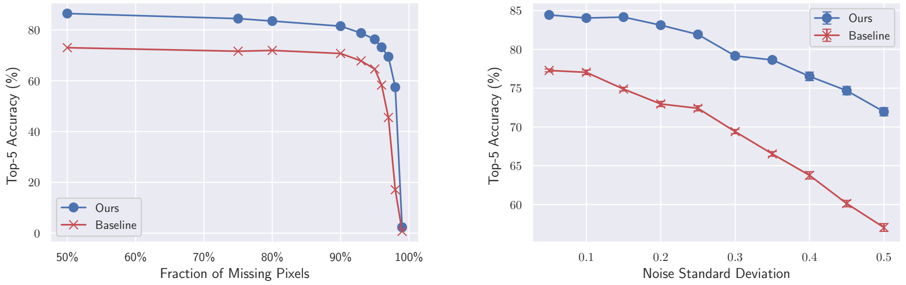

Figure 9: **Top-5 accuracies for varying noise levels on unseen classes using label shift.** On the left, we see the model trained on 50% to 95% random masking of pixels, while on the right the model trained on Gaussian noise with standard deviation from 0.1 to 0.3. Both models are evaluated on the unseen classes, using the chosen reference classes from ImageNet100 as the targets, as shown in the label shift table. Results are averaged over 10 random instantiations of corruptions. We omit error bars in the left figure as standard error is insignificant.

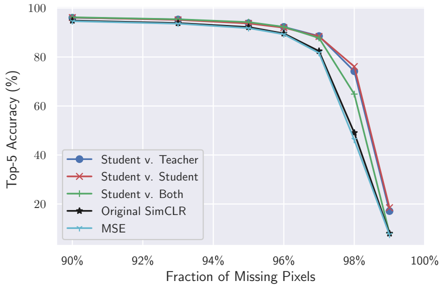

Figure 10: **Ablation study with top-5 accuracies.** We evaluate the model trained on 50% to 95% random masking of pixels, using several variants of the uniformity term in the contrastive loss. Results are averaged over 10 random instantiations of corruptions on the ImageNet-100 validation dataset. We omit error bars as standard error is insignificant.

Table 4: **Top-5 accuracy (percent) on ImageNet-100**. The best accuracy for each distortion is bolded. Each model is trained using images with a fixed type of distortion. We train our robust CLIP encoder contrastively, then fit a linear probe on the learned representations using either all or 10% of the labeled training samples. We report the mean and standard error for accuracy over 10 random instantiations of distortions on the ImageNet-100 validation dataset (Gaussian blur is deterministic, so we do not include standard error values). For Gaussian blur, n corresponds to the length of the blur kernel.

Distortion	Supervised Baseline	Ours	Ours (10% labeled data)
Random Mask 50%	93.86±0.04	**97.60±0.02**	96.57±0.06
Random Mask 75%	93.19±0.05	**96.83±0.06**	95.53±0.05
Random Mask 90%	92.12±0.05	**96.34±0.04**	95.20±0.04
Gaussian Noise $\sigma = 0.1$	95.49±0.03	**97.39±0.04**	96.11±0.03
Gaussian Noise $\sigma = 0.3$	93.00±0.05	**95.59±0.07**	93.95±0.06
Gaussian Noise $\sigma = 0.5$	90.45±0.05	**93.18±0.06**	91.14±0.06
Gaussian Blur n = 21	93.38	**96.34**	95.26
Gaussian Blur n = 37	89.42	**93.94**	91.88

Table 5: **Top-5 accuracies for transfer learning.** We fit a linear classifier for each dataset on top of the representations learned by the models from ImageNet-100. RM means the model was trained with random missing pixels, and GN means it was trained with additive Gaussian noise. Results are mean and standard errors over 10 realizations of the distortions during evaluation. The COVID X-ray dataset has two classes so there are no top-5 results.

Model	CIFAR-10	CIFAR-100	STL-10	ImageNet-100B
Baseline (RM)	98.92±0.02	84.09±0.05	99.38±0.01	88.35±0.04
Ours (RM)	**99.23±0.02**	**86.26±0.04**	**99.72±0.01**	**95.40±0.04**
Baseline (GN)	**98.76±0.02**	81.37±0.05	99.41±0.01	89.81±0.08
Ours (GN)	98.72±0.02	**81.50±0.06**	**99.48±0.01**	**94.74±0.06**

Table 6: **Area under receiver operating characteristic curve (AUC) for transfer learning on COVID X-ray data.** We fit a linear classifier for the COVID X-ray dataset on top of the representations learned by the models from ImageNet-100. RM means the model was trained with random missing pixels, and GN means it was trained with additive Gaussian noise. Results are mean and standard errors over 10 realizations of the distortions during evaluation.

Model	AUC
Baseline (RM)	0.918±0.0011
Ours (RM)	0.918±0.0010
Baseline (GN)	**0.927±0.0009**
Ours (GN)	0.896±0.0015

D.3 Comparisons with Denoising and Inpainting for Classification

An alternative way of solving the inverse problem presented in the paper is to use methods which operate directly on the pixel space of the image. Instead of training a classifier which operates on distorted images, one could try to recover the original images from the distorted version. The recovered images can then be given to a classifier trained on clean images. We compare our method with two such baselines which operate in pixel space:

1. We apply Non-Local Means (NLM) [2] denoising to images corrupted by additive Gaussian noise.
2. We use Deep Decoder [16] with default parameters and 5000 optimization steps to perform inpainting on images with random missing pixels. Deep Decoder is a method which randomly initializes an under-parameterized generative network with upsampling and 1×1 convolution layers, then optimizes over the weights of the network to fit a single distorted image. Since the network is underparameterized, it cannot fit noise very well, while its upsampling and convolution layers bias it to produce natural-looking images. The result is that the network produces a reconstructed version of the original image, despite never having been trained on any other data.

For evaluation, we corrupt images from the ImageNet-100 validation set with Gaussian noise and random pixel masks, then apply NLM and Deep Decoder, respectively. We feed the recovered images as input to a classification model trained on clean image data. The classification model we use is an ImageNet pre-trained ResNet-101 backbone with a linear classifier trained on clean ImageNet-100 images. We compare the performance of these pixel-space inverse methods with that of our method. The results for denoising are seen in Table 7 and the results for inpainting in Table 8. We can see that the inverse methods acting in pixel space are not reliable at reconstructing images for classification. In the case of denoising the accuracy degrades quickly with increasing noise, and for inpainting the accuracy is poor for all masking levels.

Table 7: **Denoising baseline.** We compare our method to a 2-step denoising baseline, where a) we denoise images with various levels of additive Gaussian noise using Non-Local Means denoising and b) we feed the images through a model trained on clean images from ImageNet-100. Results for our method are included for comparison. We perform 10 evaluation runs for each experiment and present the mean and standard error.

Noise Level	Denoising	Ours
$\sigma = 0.1$	73.49±0.08	**84.46±0.08**
$\sigma = 0.3$	56.47±0.14	**81.30±0.07**
$\sigma = 0.5$	27.64±0.14	**76.23±0.10**

Table 8: **Inpainting baseline.** We compare our method to an inpainting baseline, where a) we inpaint images with varying fractions of missing pixels using Deep Decoder [16] and b) we feed the images through a model trained on clean images from ImageNet-100. Results from the main paper are included for comparison.

Missing Pixel Fraction	Inpainting	Ours
50%	23.38	**85.87**
75%	21.89	**83.99**
90%	20.39	**82.96**

D.4 Transfering a Clean Classifier to Robust Encoders

To demonstrate the ability of our method to retrieve good representations from the teacher, we perform the following experiment. We train a robust encoder on distorted images using the method we propose. We then train a linear classifier on top of the pre-trained, non-robust CLIP backbone using clean images. Finally, we transfer this linear classifier for clean images to the robust encoder. The results can be seen in Table 9. We see that our technique achieves good results, even without finetuning the linear classifier on distorted images. This means that the representations learned by the student for distorted images are sufficiently close to those of the teacher for clean images.

Table 9: **Accuracies for applying a linear classifier trained on clean images on top of representations from a robust CLIP encoder.** We transfer the linear classifier trained on clean images on top of the representations learned by our contrastive technique. We evaluate on the same noises used during training of the robust encoder. Best scores are not bolded since we intend to illustrate a quality of our method instead of comparing two techniques.

Distortion	Clean Linear Classifier	Ours
Random Mask 50%	80.1	**85.87**
Random Mask 75%	76.4	**83.99**
Random Mask 90%	78.1	**82.96**
Gaussian Noise $\sigma = 0.1$	77.2	**84.46**
Gaussian Noise $\sigma = 0.3$	75.7	**81.30**
Gaussian Noise $\sigma = 0.5$	70.5	**76.23**
Gaussian Blur n = 21	78.3	**83.24**
Gaussian Blur n = 37	73.1	**77.80**

D.5 Decreasing Noise Levels

Extending the results from Section 4.2, we evaluate both the baseline and our robust encoder in the setting where the noise levels seen during testing are lower than those seen during training. The results can be seen in Table 10, and they are in accord with the rest of our observations: accuracy for both models is higher (due to lower noise), and our method still outperforms the baseline.

Table 10: **Evaluation of our method on lower noise levels during testing.** We extend the results from Section 4.2 to incorporate the case where the noise is lower during test time than that seen during training time. We see that our method still outperforms the baseline.

Distortion	Baseline	Ours
Random Mask 30%	76.75 ± 0.05	**86.06 ± 0.06**
Random Mask 35%	76.89 ± 0.07	**86.04 ± 0.05**
Random Mask 40%	76.93 ± 0.04	**86.11 ± 0.06**
Random Mask 45%	77.05 ± 0.08	**86.06 ± 0.05**
Gaussian Noise $\sigma = 0.02$	77.92 ± 0.02	**85.71 ± 0.02**
Gaussian Noise $\sigma = 0.04$	80.04 ± 0.05	**86.13 ± 0.04**
Gaussian Noise $\sigma = 0.06$	80.95 ± 0.05	**86.40 ± 0.04**
Gaussian Noise $\sigma = 0.08$	80.98 ± 0.04	**86.25 ± 0.07**

D.6 Baseline Model Pretrained on ImageNet-100

We examine a variant of our baseline model, where instead of using a ResNet pretrained on the full ImageNet dataset, we instead train our ResNet on ImageNet-100 to get a good classifier on the clean images, and then use that as a starting point for our baseline model. More specifically, the chosen architecture is again ResNet-101, trained in a supervised fashion for clean images for 90 epochs, with an SGD optimizer, a learning rate of 0.1, momentum of 0.9 and batch size of 256. [2] These results can be seen in Tables 11 and 12. We can see that these results are comparable to (and in most cases worse than) our original baseline. This is to be expected, since the model which is initialized with full ImageNet weights was trained on roughly 10 times more data than this new model. In any case, results are still worse than our method, which means that the latter is capable of outperforming the stronger of the two baselines.

---

[2] These are the default hyperparameters for full ImageNet training as in the official pytorch examples repository, found here: `https://github.com/pytorch/examples/tree/master/imagenet`.

Table 11: **Comparison of initial weights for supervised baseline, for the fixed noise experiment.** Training of the supervised baseline can be done starting from a ResNet-101 which was trained on the full ImageNet, or from one which was trained on only ImageNet-100. We can see that the first choice, which is the one used in the rest of this paper, is the better of the two. In any case, both baselines provide worse results than our method (compare to Table 1).

Distortion	Baseline Initial Weights From ImageNet-100	Baseline Initial Weights From Full ImageNet
Random Mask 50%	76.48 ± 0.02	**77.53 ± 0.06**
Random Mask 75%	74.38 ± 0.07	**75.68 ± 0.06**
Random Mask 90%	70.77 ± 0.10	**74.12 ± 0.09**
Gaussian Noise $\sigma = 0.1$	78.95 ± 0.06	**82.23 ± 0.04**
Gaussian Noise $\sigma = 0.3$	74.03 ± 0.08	**75.78 ± 0.08**
Gaussian Noise $\sigma = 0.5$	69.34 ± 0.09	**71.43 ± 0.14**
Gaussian Blur n = 21	72.70	**76.40**
Gaussian Blur n = 37	67.26	**68.94**

Table 12: **Comparison of initial weights for supervised baseline, for the varying noise level experiment.** Similar to Table 11, we compare the two choices for the baseline for the experiment with varying noise levels. Again, training with the full ImageNet dataset provides a stronger baseline in most cases (compare with Figure 4).

Distortion	Baseline Initial Weights From ImageNet-100	Baseline Initial Weights From Full ImageNet
Random Mask 96%	**57.21 ± 0.11**	56.80 ± 0.01
Random Mask 97%	**41.58 ± 0.07**	39.28 ± 0.15
Random Mask 98%	10.84 ± 0.09	**11.01 ± 0.12**
Random Mask 99%	1.12 ± 0.02	**2.03 ± 0.12**
Gaussian Noise $\sigma = 0.35$	71.71 ± 0.17	**72.34 ± 0.12**
Gaussian Noise $\sigma = 0.4$	68.40 ± 0.09	**69.10 ± 0.12**
Gaussian Noise $\sigma = 0.45$	62.75 ± 0.13	**65.62 ± 0.12**
Gaussian Noise $\sigma = 0.5$	54.16 ± 0.10	**61.13 ± 0.19**

D.7 Using original representations on distorted images

To understand how much we can gain from performing contrastive training to make image representations more robust, we would like to see how well the original CLIP network performs for classifying distorted images. We train a linear classifier on top of the pre-trained, non-robust CLIP backbone using distorted images (i.e. the network has not been trained with a contrastive step). We also train a linear classifier on top of the ImageNet pre-trained ResNet-101 backbone using distorted images. We present the results in Table 13. Clearly, both CLIP and ImageNet pre-trained ResNet-101 produce poor representations for distorted images resulting in low classification accuracy. These results highlight the need for a training procedure to make these pre-trained representations more robust.

Table 13: **Learning classifier on top of original representations.** Here, the linear classifier is trained on distorted images, using the original representations from CLIP and from ResNet-101 pre-trained on ImageNet. We can see that this deteriorates the results in both cases, when compared to Table 1. We do not bold best results as this experiment does not attempt to compare these two methods.

Distortion	Original CLIP Representations	Clean ResNet-101 Representations
Random Mask 50%	41.30 ± 0.06	46.30 ± 0.04
Random Mask 75%	24.00 ± 0.07	30.40 ± 0.04
Random Mask 90%	14.50 ± 0.10	23.10 ± 0.07
Gaussian Noise $\sigma = 0.1$	75.20 ± 0.06	74.50 ± 0.05
Gaussian Noise $\sigma = 0.3$	25.10 ± 0.07	50.80 ± 0.06
Gaussian Noise $\sigma = 0.5$	7.70 ± 0.09	25.20 ± 0.09
Gaussian Blur n = 21	51.20	65.80
Gaussian Blur n = 37	22.30	45.40

D.8 Experiments on ImageNet-100C

As a final benchmark, we compare our methods on a subset of ImageNet-C [18] with the same classes as those of ImageNet-100, henceforth referred to as ImageNet-100C. We compare two models:

- The first is a baseline ResNet-101, pretrained on ImageNet and finetuned on clean images of ImageNet-100.
- The second is a version of our student encoder, which is initialized from CLIP and is trained on distorted images. On these images, Gaussian blurring with $\sigma \in [1, 5]$ and Gaussian additive noise with $\sigma \in [0.05, 0.5]$ are applied, independently with probability 0.8 each.

Results can be seen in Table 14. The values for the methods on each corruption type are top-1 accuracies averaged across 5 levels of corruption. We can see here that our model does not get results comparable to the baseline. This can be explained by a limitation of our current work, in that we rely on student representations matching the teacher representations. If the *type* of noise is altered for the student, then it is difficult for it to match the teacher representations, which are fixed. Indeed, altering the type of noise on an image is expected to greatly affect its representation. Thus, at its present iteration, our technique relies on some prior knowledge about the *type* of distortion encountered. Possible ways around this, such as also finetuning the teacher, are left for consideration in future work.

Table 14: **ImageNet-100C results.** We present the mean top-1 accuracies across 5 corruption levels for each corruption type. We do not present mean corruption error (mCE) as this is computed with respect to performance on full ImageNet-C, while we compute on a 100-class subset of ImageNet-C.

	Corruption	Supervised Baseline	Ours
Blur	Defocus	**61.76**	56.88
	Glass	49.15	**50.52**
	Motion	**59.38**	44.37
	Zoom	**59.00**	50.69
Digital	Contrast	**58.61**	43.69
	Elastic	**67.85**	60.40
	JPEG	**77.75**	61.08
	Pixelate	73.06	**75.65**
Noise	Gaussian	55.83	**56.81**
	Impulse	51.05	**54.11**
	Shot	53.76	**56.74**
Weather	Brightness	**86.01**	72.57
	Fog	**67.56**	54.20
	Frost	**59.58**	51.70
	Snow	**53.97**	39.80
Extra	Gaussian Blur	**64.31**	62.66
	Saturate	**82.16**	65.95
	Spatter	**71.9**	59.52
	Speckle	62.65	**66.28**

D.9 ImageNet-100 Classes

To train our model and the baseline, we use a randomly-chosen 100-class subset of the original ImageNet dataset. This subset is the same as used in [39]. We present the `wnid` of each of the classes in the subset in Table 15

D.10 ImageNet-100B Classes

For the transfer learning experiments, we select a random subset of 100 classes from ImageNet which are mutually exclusive with the classes found in ImageNet-100. We term this subset ImageNet-100B, and present the `wnid` of each of the classes used in this subset in Table 16.

Table 15: **List of ImageNet-100 Classes.** We present the `wnid` of each of the classes used in ImageNet-100. These classes are randomly sampled from the original ImageNet dataset and are the same classes used in [39].

ImageNet-100 Classes				
n02869837	n02086910	n03785016	n02483362	n03837869
n01749939	n02859443	n03764736	n04127249	n03494278
n02488291	n13040303	n03775546	n02089973	n04136333
n02107142	n03594734	n02087046	n03017168	n03794056
n13037406	n02085620	n07836838	n02093428	n03492542
n02091831	n02099849	n04099969	n02804414	n02018207
n04517823	n01558993	n04592741	n02396427	n04067472
n04589890	n04493381	n03891251	n04418357	n03930630
n03062245	n02109047	n02701002	n02172182	n03584829
n01773797	n04111531	n03379051	n01729322	n02123045
n01735189	n02877765	n02259212	n02113978	n04229816
n07831146	n04429376	n07715103	n03787032	n02100583
n07753275	n02009229	n03947888	n02089867	n03642806
n03085013	n01978455	n04026417	n02119022	n04336792
n04485082	n02106550	n02326432	n03777754	n03259280
n02105505	n01820546	n03637318	n04238763	n02116738
n01983481	n01692333	n01980166	n02231487	n02108089
n02788148	n07714571	n02113799	n03032252	n03424325
n03530642	n02974003	n02086240	n02138441	n01855672
n04435653	n02114855	n03903868	n02104029	n02090622

Table 16: **List of ImageNet-100B Classes.** We present the `wnid` of each of the classes used in ImageNet-100B. These classes are randomly sampled from the original ImageNet dataset and are mutually exclusive with the classes in ImageNet-100.

ImageNet-100B Classes				
n02088364	n02840245	n04258138	n03670208	n02013706
n03000134	n01688243	n02280649	n03483316	n02797295
n03544143	n03920288	n02492660	n02777292	n04366367
n03388043	n02488702	n03782006	n03602883	n03857828
n02165105	n03884397	n03495258	n03982430	n04243546
n02321529	n07745940	n04254120	n02808440	n03891332
n01819313	n01484850	n02391049	n03207743	n03796401
n03187595	n04147183	n04254777	n02096177	n03314780
n01667114	n04356056	n07716906	n01742172	n04039381
n02097130	n01644900	n03888605	n03792782	n01498041
n02104365	n02132136	n01843065	n01795545	n01990800
n02279972	n02999410	n02643566	n03534580	n03976657
n12985857	n02457408	n04515003	n03814906	n02107683
n01773549	n04540053	n03125729	n02342885	n02229544
n12057211	n02776631	n04179913	n03692522	n03063599
n02791270	n02107574	n03788365	n03272010	n03127747
n01491361	n02930766	n02098286	n09468604	n03179701
n02169497	n04263257	n02109525	n03720891	n03016953
n02793495	n03724870	n02966193	n03717622	n09193705
n02281787	n04081281	n03929660	n02177972	n04033901

E Visualization of Distortions

In this section, we present several examples of the fixed distortions we apply during training and testing. All images are taken from the ImageNet-100 validation set.

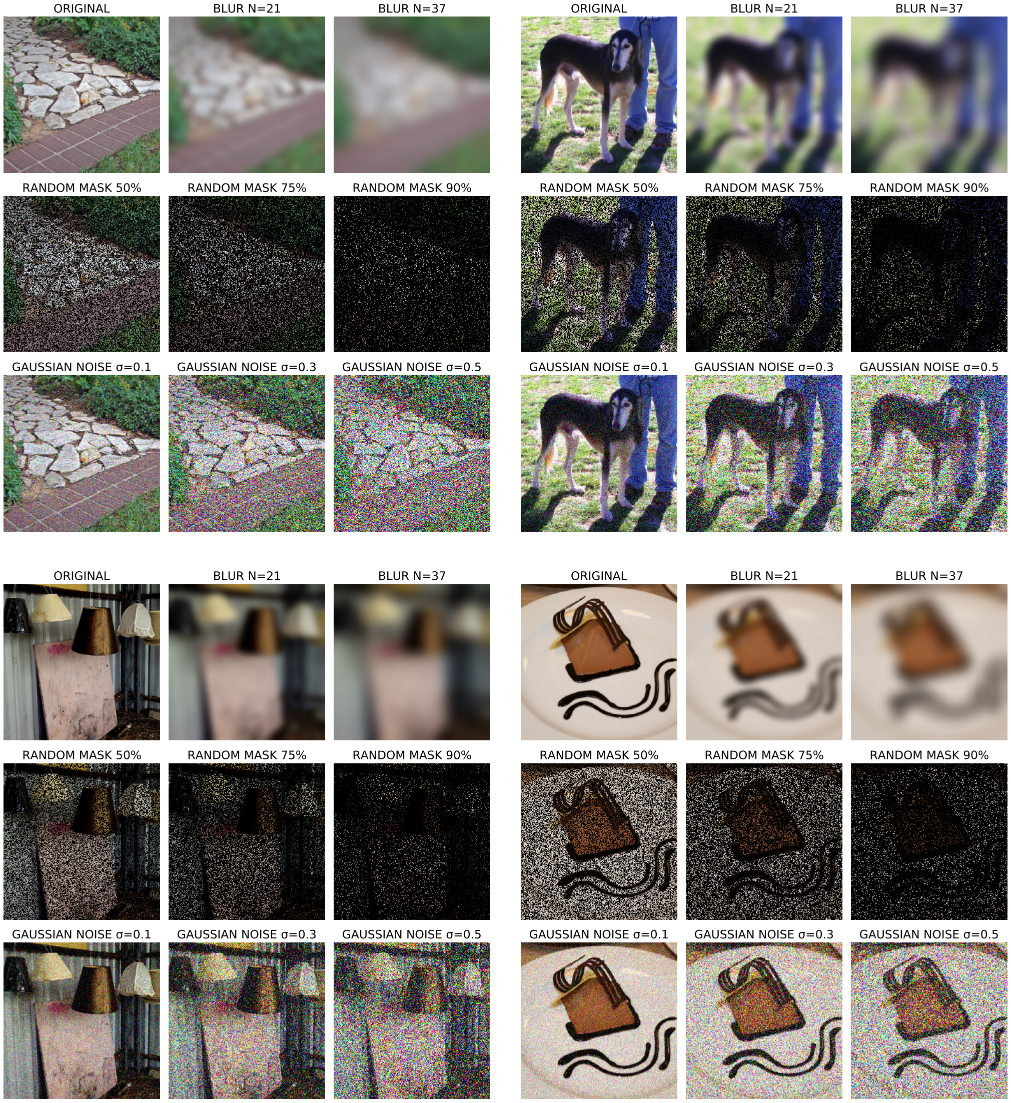

Figure 11: **Visualization of the various fixed distortions we test.**