

Trustworthy ML: Robustness and Foresight

Saurav Kadavath



Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-245

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-245.html>

December 1, 2021

Copyright © 2021, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

The work presented throughout the entirety of this report is joint work with others, including Dan Hendrycks, Mantas Mazeika, Steven Basart, Norman Mu, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Collin Burns, Akul Arora, Eric Tang, Jacob Steinhardt, Dawn Song, and Justin Gilmer. I would like to especially thank Dan Hendrycks for his support and mentorship over the past 2.5 years.

Trustworthy ML: Robustness and Foresight

Saurav Kadavath

May 2021

Trustworthy ML: Robustness and Foresight

by Saurav Kadavath

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II.**

Approval for the Report and Comprehensive Examination:

Committee:

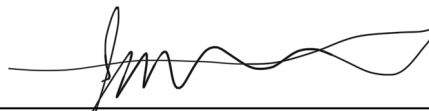


Professor Dawn Song
Research Advisor

5/11/2021

(Date)

* * * * *



Professor Jacob Steinhardt
Second Reader

05 / 21 / 2021

(Date)

To my Family, for their endless love and support.

The work presented throughout the entirety of this report is joint work with others, including Dan Hendrycks, Mantas Mazeika, Steven Basart, Norman Mu, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Collin Burns, Akul Arora, Eric Tang, Jacob Steinhardt, Dawn Song, and Justin Gilmer. [\[1\]](#) [\[2\]](#) [\[3\]](#)

I would like to especially thank Dan Hendrycks for his support and mentorship over the past 2.5 years.

Copyright © 2021, The author(s). All rights reserved.

¹ *Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty* (arxiv.org/abs/1906.12340)

² *The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization* (arxiv.org/abs/2006.16241)

³ *Measuring Mathematical Problem Solving With the MATH Dataset* (arxiv.org/abs/2103.03874)

Contents

Contents	4
List of Figures	6
List of Tables	7
Abstract	8
Part 1. Robustness	10
1 Distribution Shift in Computer Vision	11
1.1 ImageNet-R	11
1.2 DeepAugment	12
1.2.1 Noise2Net	13
1.3 Experiments	13
1.4 Results	15
1.4.0.1 ImageNet-R.	16
1.4.0.2 Ablations.	17
1.4.0.3 ImageNet-C.	18
2 Self-Supervised Learning and Model Robustness & Uncertainty	19
2.1 Self-Supervision	19
2.2 Robustness to Adversarial Perturbations	20
2.2.1 Setup	20
2.2.2 Method	20
2.2.3 Results	21
2.3 Robustness to Common Corruptions	22
2.3.1 Setup	22
2.3.2 Results	22
2.4 Multi-Class Out-of-Distribution Detection	23
2.4.1 Setup	23
2.4.2 Method	23
2.4.3 Results	24
Part 2. Foresight	25
3 Mathematical Problem Solving	26
3.1 The MATH Dataset	26
3.1.1 Categorizing Problems.	27
3.1.2 Formatting	28

3.1.3	Automatically Assessing Generated Answers.	28
3.1.4	Human-Level Performance.	28
3.2	AMPS (Khan + Mathematica) Dataset	29
3.2.1	Khan Academy.	29
3.2.2	Mathematica.	29
3.3	Experiments	30
3.3.1	Setup	30
3.3.2	Results	30
3.3.2.1	Results vs. Model Size	30
3.3.2.2	Results vs. Problem Difficulty	31
3.3.2.3	The effect of AMPS Pretraining	31
3.3.2.4	Error Detection	32
3.3.2.5	The Benefits of MATH Solutions	32
	Bibliography	35

List of Figures

1.1	ImageNet-Renditions (ImageNet-R) contains 30,000 images of ImageNet objects with different textures and styles. This figure shows only a portion of ImageNet-R’s numerous rendition styles. The rendition styles (e.g., “Toy”) are for clarity and are <i>not</i> ImageNet-R’s classes; ImageNet-R’s classes are a subset of 200 ImageNet classes. ImageNet-R emphasizes shape over texture.	12
1.2	Pythonic Pseudocode for DeepAugment	14
1.3	DeepAugment examples preserve semantics, are data-dependent, and are far more visually diverse than augmentations such as rotations.	15
1.4	Example outputs of Noise2Net for different values of ε . Note $\varepsilon = 0$ is the original image.	16
1.5	Parallel augmentation with Noise2Net. We collapse batches to the channel dimension to ensure that different transformations are applied to each image in the batch. Feeding images into the network in the standard way would result in the same augmentation being applied to each image, which is undesirable. The function $f_{\Theta}(x)$ is a Res2Net block with all convolutions replaced with grouped convolutions.	16
1.6	The left figure shows accuracy as a function of corruption severity. In the right figure, we show ImageNet accuracy and ImageNet-C accuracy. Previous architectural advances slowly translate to ImageNet-C performance improvements, but the DeepAugment+AugMix robustness intervention on a ResNet-50 approximately yields a 19% accuracy improvement.	18
2.1	The effect of attack strength on a $\varepsilon = 8/255$ adversarially trained model. The attack strengths are $\varepsilon \in \{4/255, 5/255, \dots, 10/255\}$. Since the accuracy gap widens as ε increases, self-supervision’s benefits are masked when observing the clean accuracy alone.	21
2.2	A comparison of the accuracy of usual training compared to training with auxiliary rotation self-supervision on the nineteen CIFAR-10-C corruptions. Each bar represents an average over all five corruption strengths for a given corruption type.	22
2.3	OOD detection performance of the maximum softmax probability baseline and our method using self-supervision. Full results are in Chapter ??.	24
3.1	Previous work is based on formal theorem provers or straightforward plug-and-chug problems. Our dataset, MATH, has competition mathematics problems with step-by-step solutions written in L ^A T _E X and natural language. Models are tasked with generating tokens to construct the final (boxed) answer.	27
3.2	Problems that are more difficult for humans are also more difficult for GPT-2.	31
3.3	Accuracy on MATH per subject per difficulty level, with GPT-2 1.5B and GPT-2 0.1B.	31
3.5	Histogram of the answer confidences of GPT-2 1.5B. The incorrect answer histogram is translucent and overlays the correct answer histogram. GPT-2 1.5B is overconfident and has an AUROC of 68.8%, indicating that it is not good at detecting errors from confidence alone. Confidences below 0.75 are omitted for ease of visualization.	33
3.6	Models conditioned on most of a problem’s step-by-step solution can often understand the solution to predict the final answer. ‘99%’ of a solution is all the solution text before the final answer. Not all solutions have an answer that is immediate from the preceding solution text.	33

List of Tables

1.1	ImageNet-200 and ImageNet-Renditions error rates. ImageNet-200 uses the same 200 classes as ImageNet-R. DeepAugment+AugMix improves over the baseline by over 10 percentage points. ImageNet-21K Pretraining tests Pretraining . Style Transfer, AugMix, and DeepAugment test Diverse Data Augmentation in contrast to simpler noise augmentations such as ℓ_∞ Adversarial Noise and Speckle Noise. While there remains much room for improvement, results indicate that progress on ImageNet-R is tractable. ImageNet-21K and WSL Pretraining test the Pretraining hypothesis, and here pretraining gives mixed benefits. and ResNeXt-101 32×8d tests the Larger Models hypothesis, and this helps. Other methods augment data, and Style Transfer, AugMix, and DeepAugment provide support for the Diverse Data Augmentation hypothesis.	17
1.2	DeepAugment (EDSR, CAE, Noise2Net) ablations on ImageNet-200 and ImageNet-Renditions. .	17
1.3	Clean Error, Corruption Error (CE), and mean CE (mCE) values for DeepAugment ablations on ImageNet-C. The mCE value is computed by averaging across all 15 CE values.	18
2.1	Results for our defense. All results use $\varepsilon = 8.0/255$. For 20-step adversaries $\alpha = 2.0/255$, and for 100-step adversaries $\alpha = 0.3/255$. More steps do not change results, so the attacks converge. Self-supervision through rotations provides large gains over standard adversarial training.	21
2.2	Out-of-distribution example detection results for the maximum softmax probability (MSP) baseline and our rotation method. All results are percentages and the average result of 5 runs.	24
3.1	MATH accuracies across subjects for GPT-2 and <i>few-shot</i> GPT-3 models. The character ‘B’ denotes the number of parameters in billions. The gray text indicates the <i>relative</i> improvement over the 0.1B baseline. All GPT-2 models pretrain on AMPS, and all values are percentages. A 15× increase in model parameters increased accuracy by 1.5%, a 28% relative improvement. Model accuracy is increasing very slowly, so much future research is needed.	29

Abstract

Over the last decade, AI has empowered people, businesses, and governments to create some of the most impressive products and solve some of the most challenging problems - self-driving cars, virtual assistants, and pharmaceutical drug discovery to name a few. It is clear that AI will have a long-lasting impact on humanity. In this context, our goal is to understand how we can ensure that this impact is as positive as possible. This is a very broad problem, but we can tease it apart into several smaller goals that can fit under one of three umbrellas: (1) Making AI competent (e.g. making models robust, reliable, and understand humans), (2) Making AI aligned, and (3) Coping with the effects of AI ([Paul Christiano: Current Work in AI Alignment](#) [2020](#)).

Part 1. In the first part of this report, we work on making AI competent, focusing on computer vision and making progress in improving model robustness. In Chapter 1, we develop a novel data augmentation strategy, called DeepAugment. Whereas previous data augmentation strategies relied on simple image transformations such as rotations, color shift, shearing, resizing, etc..., DeepAugment creates augmentations using a single forward pass through a pretrained and perturbed image-to-image neural network. This allows us to generate augmented images using transformations that are much more complex than the simple transforms used in previous work. We extend DeepAugment with Noise2Net, a method of creating image augmentations using a completely randomly initialized neural network.

We also introduce the ImageNet-Renditions benchmark, a new evaluation benchmark for ImageNet models. ImageNet-Renditions (ImageNet-R) contains 30,000 test set images of various renditions (e.g., paintings, embroidery, etc.) of ImageNet objects from 200 out of the 1000 total classes. We show DeepAugment and Noise2Net to be effective in improving ImageNet-R performance.

Part 2. In Part 2, we work develop new datasets and benchmarks to understand the performance of large state-of-the-art language models at various reasoning tasks. It is important for the AI community to carefully monitor the state of AI at advanced reasoning tasks, since models approaching human performance here could have profound social and economic implications. So far, modern large-scale language models such as GPT (Brown et al. [2020](#)), T5 (Raffel et al. [2020](#)), BERT (Devlin et al. [2019](#)), etc. have demonstrated impressive capabilities across a variety of text-based tasks. Performance on a wide variety of benchmarks have been shown to be growing steadily as model sizes increase (A. Wang et al. [2019](#); Zellers et al. [2019](#); Huang et al. [2019](#); Bisk et al. [2019](#); Hendrycks, Basart, et al. [2020](#); Hendrycks, Burns, Basart, Zou, et al. [2021](#); Hendrycks, Burns, Basart, Critch, et al. [2021](#)).

We introduce MATH, a dataset and benchmark for mathematical problem solving. Mathematics problems are valuable tests for *problem-solving ability*: the ability to analyze a problem, pick out good heuristics from a large set of possibilities, and chain them together to produce an answer. This contrasts with plug-and-chug calculations, a skill which ML models can already exhibit (Henighan et al. [2020](#)). Additionally, our benchmark does not involve multiple-choice answers - models are trained to output the complete answer on its own.

We show that simply scaling up models aggressively probably will not give us solutions for MATH. Our

results suggest that further advancements need to be made in order to create models that can reason at a human's level.

We leave the 2nd question of making AI aligned to future work. Code and datasets are released for each chapter at the following locations:

1. **Distribution Shift in Computer Vision:**
<https://github.com/hendrycks/imagenet-r>
2. **Self-Supervised Learning to Improve Robustness and Uncertainty:**
<https://github.com/hendrycks/ss-ood>
3. **Mathematical Problem Solving:**
<https://github.com/hendrycks/math>

FIRST PART

ROBUSTNESS

Chapter 1

Distribution Shift in Computer Vision

Deep Learning has exploded in popularity over the last decade, as a solution to a wide variety of problems in computer vision, NLP, speech recognition, reinforcement learning, and more.

In order to ensure that these models are competent as they are deployed to a wider variety of situations, one of the central problems that must be solved revolves around how to make these models generalizable and robust to unforeseen distribution shifts that may be encountered in the wild. In this chapter, we focus specifically on methods to benchmark and improve performance of deep neural networks used for image classification under various distribution shifts.

While the research community must create robust models that generalize to new scenarios, the robustness literature (Dodge and Karam [2017](#); Geirhos, Jacobsen, et al. [2020](#)) lacks consensus on evaluation benchmarks and contains many dissonant hypotheses. (Hendrycks, Liu, et al. [2020](#)) find that many recent language models are already robust to many forms of distribution shift, while (Yin et al. [2019](#)) and (Geirhos, Rubisch, et al. [2019](#)) find that vision models are largely fragile and argue that data augmentation offers one solution. In contrast, (Taori et al. [2020](#)) provide results suggesting that using pretraining and improving in-distribution test set accuracy improve natural robustness, whereas other methods do not.

In this chapter, we develop a novel method of data augmentation that improves model robustness and introduce a new benchmark for distribution shift. We also articulate and study four robustness hypotheses. These hypotheses are as follows.

- **Larger Models:** Increasing model size improves robustness (Hendrycks and Dietterich [2019](#); C. Xie and Yuille [2020](#)).
- **Diverse Data Augmentation:** Robustness can increase through data augmentation (Yin et al. [2019](#)).
- **Pretraining:** Pretraining on larger and more diverse datasets improves robustness (Orhan [2019](#); Hendrycks, Lee, and Mazeika [2019](#)).
- **Texture Bias:** Convolutional networks are biased towards texture, which harms robustness (Geirhos, Rubisch, et al. [2019](#)).

1.1 ImageNet-R

While current classifiers can learn some aspects of an object’s shape (Mordvintsev, Olah, and Tyka [2015](#)), they nonetheless rely heavily on natural textural cues (Geirhos, Rubisch, et al. [2019](#)). In contrast, human vision can process abstract visual renditions. For example, humans can recognize visual scenes from line drawings as quickly and accurately as they can from photographs (Biederman and Ju [1988](#)). Even some

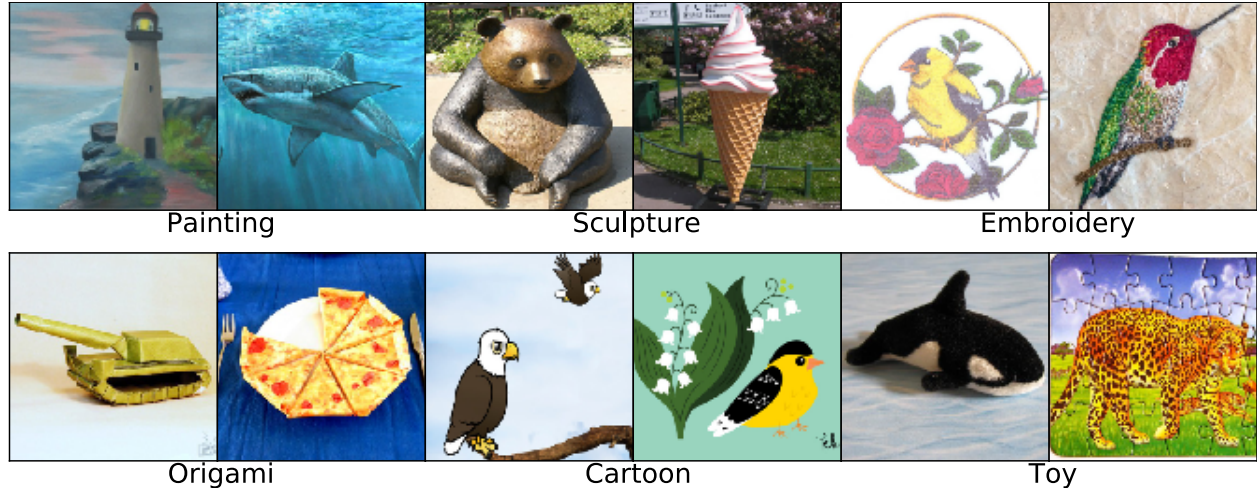


Figure 1.1: ImageNet-Renditions (ImageNet-R) contains 30,000 images of ImageNet objects with different textures and styles. This figure shows only a portion of ImageNet-R’s numerous rendition styles. The rendition styles (e.g., “Toy”) are for clarity and are *not* ImageNet-R’s classes; ImageNet-R’s classes are a subset of 200 ImageNet classes. ImageNet-R emphasizes shape over texture.

primates species have demonstrated the ability to recognize shape through line drawings (Itakura [1994], Tanaka [2006]).

To measure generalization to various abstract visual renditions, and to help us evaluate and better understand the **Texture Bias** hypothesis, we create the ImageNet-Rendition (ImageNet-R) dataset. ImageNet-R contains various artistic renditions of object classes from the original ImageNet dataset. Note the original ImageNet dataset discouraged such images since annotators were instructed to collect “photos only, no painting, no drawings, etc.” (Deng [2012]). We do the opposite.

Data Collection. ImageNet-R contains 30,000 image renditions for 200 ImageNet classes. We choose a subset of the ImageNet-1K classes, following (Hendrycks, Zhao, et al. [2019]), for several reasons. A handful ImageNet classes already have many renditions, such as “triceratops.” We also choose a subset so that model misclassifications are egregious and to reduce label noise. The 200 class subset was also chosen based on rendition prevalence, as “strawberry” renditions were easier to obtain than “radiator” renditions. Were we to use all 1,000 ImageNet classes, annotators would be pressed to distinguish between Norwich terrier renditions as Norfolk terrier renditions, which is difficult. ImageNet-R also includes the line drawings from (H. Wang et al. [2019]), excluding horizontally mirrored duplicate images, pitch black images, and images from the incorrectly collected “pirate ship” class.

In the following sections, ImageNet-R is used as a test set for models trained on the standard ImageNet dataset. Since this framework induces a distribution shift between train time and test time, it helps us understand and benchmark the generalization capabilities of various models and training methods.

1.2 DeepAugment

In order to further explore the **Diverse Data Augmentation** hypothesis, we introduce a new data augmentation technique. Whereas most previous data augmentations techniques use simple augmentation primitives applied to the raw image itself (e.g. rotations, scaling, color shift, etc.), we introduce DeepAugment, which distorts images by perturbing internal representations of deep networks.

DeepAugment works by passing a clean image through an image-to-image network and introducing several

perturbations during the forward pass. These perturbations are randomly sampled from a set of manually designed functions and applied to the network weights and to the feed-forward signal at random layers. For example, our set of perturbations includes zeroing, negating, convolving, transposing, applying activation functions, and more. This setup generates semantically consistent images with unique and diverse distortions Figure 1.3. Although our set of perturbations is designed with random operations, we show that DeepAugment still outperforms other methods on benchmarks such as ImageNet-C and ImageNet-R. We provide pseudocode in Figure 1.2.

For our experiments, we specifically use the CAE (Theis et al. 2017) and EDSR (Lim et al. 2017) architectures as the basis for DeepAugment. CAE is an autoencoder architecture, and EDSR is a superresolution architecture. These two architectures show the DeepAugment approach works with different architectures. Each clean image in the original dataset and passed through the network and is thereby stochastically distorted, resulting in two distorted versions of the clean dataset (one for CAE and one for EDSR). We then train on the augmented and clean data simultaneously and call this approach DeepAugment.

1.2.1 Noise2Net

In this section we explore a variant of DeepAugment, called Noise2Net, where we use randomly initialized image-to-image networks to generate diverse image augmentations.

In Noise2Net, the architecture and weights are randomly sampled. Noise2Net is the composition of several residual blocks: $\text{Block}(x) = x + \varepsilon \cdot f_{\Theta}(x)$, where Θ is randomly initialized and ε is a parameter that controls the strength of the augmentation. For all our experiments, we use 4 Res2Net blocks (Gao et al. 2019) and $\varepsilon \sim U(0.375, 0.75)$. The weights of Noise2Net are resampled at every minibatch, and the dilation and kernel sizes of all the convolutions used in Noise2Net are randomly sampled every epoch. Hence, Noise2Net augments an image to an augmented image by processing the image through a randomly sampled network with random weights.

Recall that in the case of EDSR and CAE, we used networks to generate a static dataset, and then we trained normally on that static dataset. This setup could not be done on-the-fly. That is because we fed in one example at a time with EDSR and CAE. If we pass the entire minibatch through EDSR or CAE, we will end up applying the same augmentation to all images in the minibatch, reducing stochasticity and augmentation diversity. In contrast, Noise2Net enables us to process batches of images on-the-fly and obviates the need for creating a static augmented dataset.

In Noise2Net, each example in a minibatch is processed differently in parallel, so we generate diverse augmentations in real-time. To make this possible, we use grouped convolutions. A grouped convolution with number of groups = N will take a set of kN channels as input, and apply N independent convolutions on channels $\{1, \dots, k\}, \{k+1, \dots, 2k\}, \dots, \{(N-1)k+1, \dots, Nk\}$. Given a minibatch of size B , we can apply a randomly initialized grouped convolution with $N = B$ groups in order to apply a different random convolutional filter to each element in the batch in a single forward pass. By replacing all the convolutions in each Res2Net block with a grouped convolution and randomly initializing network weights, we arrive at Noise2Net, a variant of DeepAugment. See Figure 1.5 for a high-level overview of Noise2Net and Figure 1.4 for sample outputs.

1.3 Experiments

In this section we briefly describe the evaluated models, pretraining techniques, self-attention mechanisms, data augmentation methods, and note various implementation details.

Model Architectures and Sizes. Most experiments are evaluated on a standard ResNet-50 model (He et al. 2015). Model size evaluations use ResNets or ResNeXts (S. Xie et al. 2016) of varying sizes to help us evaluate the **Larger Models** hypothesis.

```

1  def main():
2      net.apply_weights(deepAugment_getNetwork()) # EDSR, CAE, ...
3      for image in dataset: # May be the ImageNet training set
4          if np.random.uniform() < 0.05: # Arbitrary refresh prob
5              net.apply_weights(deepAugment_getNetwork())
6              new_image = net.deepAugment_forwardPass(image)
7
8  def deepAugment_getNetwork():
9      weights = load_clean_weights()
10     weight_distortions = sample_weight_distortions()
11     for d in weight_distortions:
12         weights = apply_distortion(d, weights)
13     return weights
14
15 def sample_weight_distortions():
16     distortions = [
17         negate_weights,
18         zero_weights,
19         flip_transpose_weights,
20         ...
21     ]
22
23     return random_subset(distortions)
24
25 def sample_signal_distortions():
26     distortions = [
27         gelu,
28         negate_signal_random_mask,
29         flip_signal,
30         ...
31     ]
32
33     return random_subset(distortions)
34
35
36 class Network():
37     def apply_weights(weights):
38         ... # Apply given weight tensors to network
39
40     # Clean forward pass. Compare to deepAugment_forwardPass()
41     def clean_forwardPass(X):
42         X = network.block1(X)
43         X = network.block2(X)
44         ...
45         X = network.blockN(X)
46         return X
47
48     # Our forward pass. Compare to clean_forwardPass()
49     def deepAugment_forwardPass(X):
50         # Returns a list of distortions, each of which
51         # will be applied at a different layer.
52         signal_distortions = sample_signal_distortions()
53
54         X = network.block1(X)
55         apply_layer_1_distortions(X, signal_distortions)
56         X = network.block2(X)
57         apply_layer_2_distortions(X, signal_distortions)
58         ...
59         apply_layer_N-1_distortions(X, signal_distortions)
60         X = network.blockN(X)
61         apply_layer_N_distortions(X, signal_distortions)
62
63     return X

```

Figure 1.2: Pythonic Pseudocode for DeepAugment

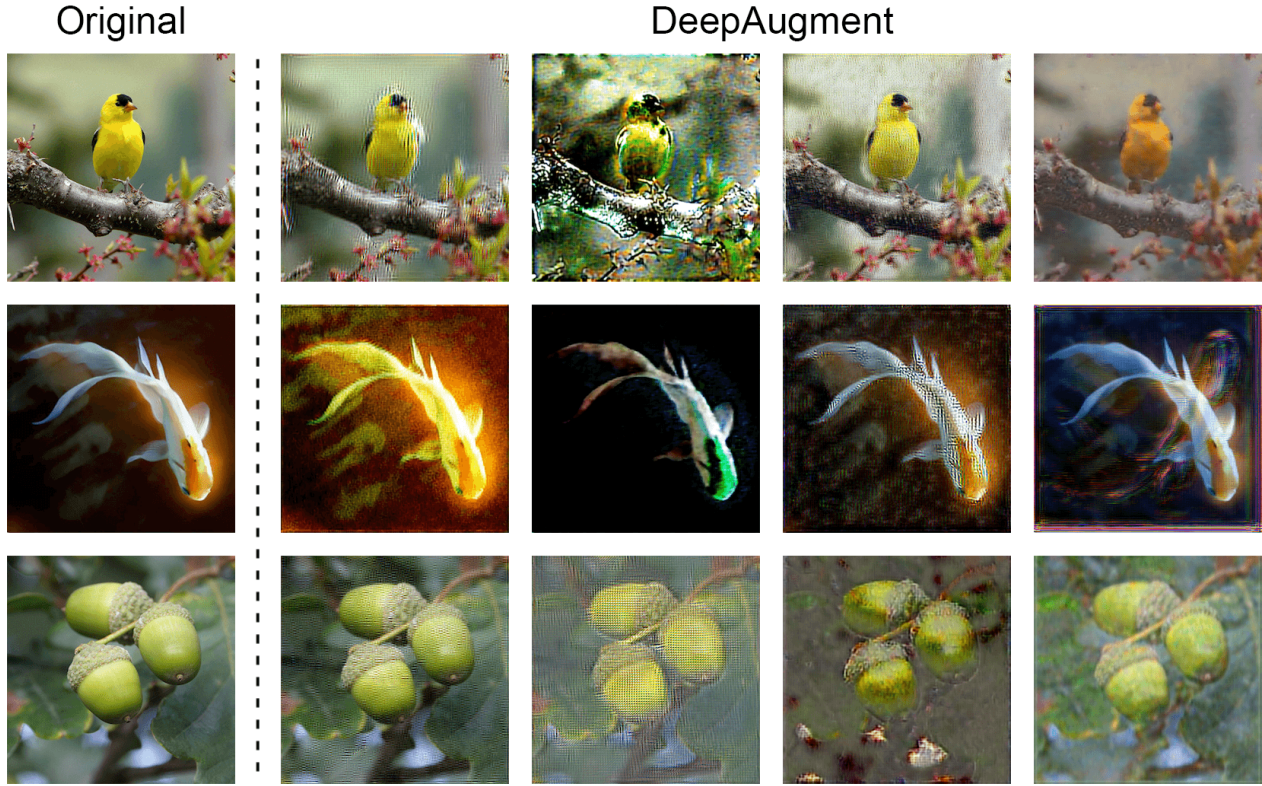


Figure 1.3: DeepAugment examples preserve semantics, are data-dependent, and are far more visually diverse than augmentations such as rotations.

Pretraining. To analyze the **Pretraining** hypothesis, we use ImageNet-21K which contains approximately 21,000 classes and approximately 14 million labeled training images, or around $10\times$ more labeled training data than ImageNet-1K. We tune (Kolesnikov et al. [2019])’s ImageNet-21K model. We also use a large pre-trained ResNeXt-101 model from (Mahajan et al. [2018]). This was pre-trained on approximately 1 billion Instagram images with hashtag labels and fine-tuned on ImageNet-1K. This Weakly Supervised Learning (WSL) pretraining strategy uses approximately $1000\times$ more labeled data.

Data Augmentation. We use Style Transfer, AugMix, and DeepAugment to analyze the **Diverse Data Augmentation** hypothesis, and we contrast their performance with simpler noise augmentations such as Speckle Noise and adversarial noise. Style transfer (Geirhos, Rubisch, et al. [2019]) uses a style transfer network to apply artwork styles to training images. We use AugMix (Hendrycks, Mu, et al. [2020]) which randomly composes simple augmentation operations (e.g., translate, posterize, solarize). DeepAugment, introduced above, distorts the weights and feedforward passes of image-to-image models to generate image augmentations. Speckle Noise data augmentation multiplies each pixel by $(1+x)$ with x sampled from a normal distribution (Rusak et al. [2020]; Hendrycks and Dietterich [2019]). We also consider adversarial training as a form of adaptive data augmentation and use the model from (Wong, Rice, and Kolter [2020]) trained against ℓ_∞ perturbations of size $\varepsilon = 4/255$.

1.4 Results

We now perform experiments using DeepAugment and other methods to test generalization capabilities on ImageNet-R and ImageNet-C.

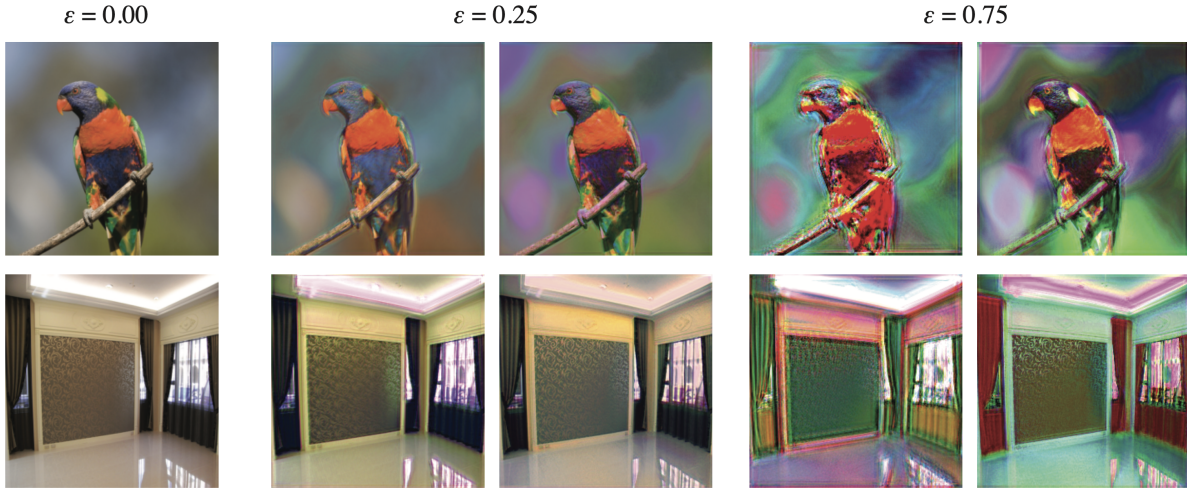


Figure 1.4: Example outputs of Noise2Net for different values of ε . Note $\varepsilon = 0$ is the original image.

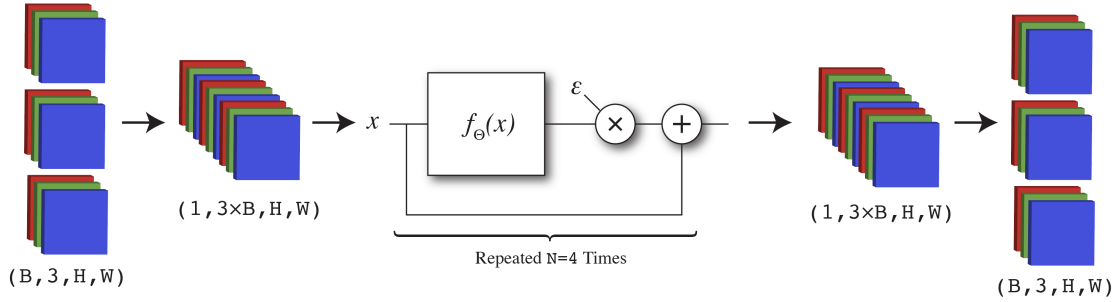


Figure 1.5: Parallel augmentation with Noise2Net. We collapse batches to the channel dimension to ensure that different transformations are applied to each image in the batch. Feeding images into the network in the standard way would result in the same augmentation being applied to each image, which is undesirable. The function $f_{\Theta}(x)$ is a Res2Net block with all convolutions replaced with grouped convolutions.

1.4.0.1 ImageNet-R.

Table 1.1 shows performance on ImageNet-R as well as on ImageNet-200 (the original ImageNet data restricted to ImageNet-R’s 200 classes). This has several implications regarding the four method-specific hypotheses. **Pretraining** with ImageNet-21K (approximately $10\times$ labeled data) hardly helps. Table 1.1 shows WSL pretraining can help, but Instagram has renditions, while ImageNet excludes them; hence we conclude comparable pretraining was ineffective. Compared to simpler data augmentation techniques such as Speckle Noise, the **Diverse Data Augmentation** techniques of Style Transfer, AugMix, and DeepAugment improve generalization. Note AugMix and DeepAugment improve in-distribution performance whereas Style transfer hurts it. Also, our new DeepAugment technique is the best standalone method with an error rate of 57.8%. Last, **Larger Models** reduce the IID/OOD gap.

DeepAugment and Noise2Net were designed to keep high-level semantic image characteristics constant, while diversifying texture. Biasing networks away from natural textures using DeepAugment and Noise2Net im-

	ImageNet-200 (%)	ImageNet-R (%)	Gap
ResNet-50 (He et al. 2015)	7.9	63.9	56.0
+ ImageNet-21K Pretraining (10× data)	7.0	62.8	55.8
+ ℓ_∞ Adversarial Training	25.1	68.6	43.5
+ Speckle Noise	8.1	62.1	54.0
+ Style Transfer	8.9	58.5	49.6
+ AugMix	7.1	58.9	51.8
+ DeepAugment	7.5	57.8	50.3
+ DeepAugment + AugMix	8.0	53.2	45.2
ResNeXt-101 32×8d (Larger Models)	6.2	57.5	51.3
+ WSL Pretraining (1000× data)	4.1	24.2	20.1
+ DeepAugment + AugMix	6.1	47.9	41.8

Table 1.1: ImageNet-200 and ImageNet-Renditions error rates. ImageNet-200 uses the same 200 classes as ImageNet-R. DeepAugment+AugMix improves over the baseline by over 10 percentage points. ImageNet-21K Pretraining tests **Pretraining**. Style Transfer, AugMix, and DeepAugment test **Diverse Data Augmentation** in contrast to simpler noise augmentations such as ℓ_∞ Adversarial Noise and Speckle Noise. While there remains much room for improvement, results indicate that progress on ImageNet-R is tractable. ImageNet-21K and WSL Pretraining test the **Pretraining** hypothesis, and here pretraining gives mixed benefits. and ResNeXt-101 32×8d tests the **Larger Models** hypothesis, and this helps. Other methods augment data, and Style Transfer, AugMix, and DeepAugment provide support for the **Diverse Data Augmentation** hypothesis.

	ImageNet-200 (%)	ImageNet-R (%)	Gap
ResNet-50	7.9	63.9	56.0
+ DeepAugment (EDSR)	7.9	60.3	55.1
+ DeepAugment (CAE)	7.6	58.5	50.9
+ DeepAugment (EDSR + CAE)	7.5	57.8	50.3
+ DeepAugment (Noise2Net)	7.2	57.6	50.4
+ DeepAugment (All 3)	7.4	56.0	48.6

Table 1.2: DeepAugment (EDSR, CAE, Noise2Net) ablations on ImageNet-200 and ImageNet-Renditions.

proved performance, so we find support for the **Texture Bias** hypothesis.

1.4.0.2 Ablations.

We run ablations on DeepAugment to understand the contributions from the EDSR, CAE, and Noise2Net models independently. Table 1.2 contains results of these experiments on ImageNet-R and Table 1.3 contains results of these experiments on ImageNet-C. In both tables, “DeepAugment (EDSR)” and “DeepAugment (CAE)” refer to experiments where we only use a single extra augmented training set (+ the standard training set), and train on those images.

We evaluate the Noise2Net variant of DeepAugment on ImageNet-R. Table 1.2 shows that it outperforms the EDSR and CAE variants of DeepAugment, even though the network architecture is randomly sampled, its weights are random, and the network is not trained. This demonstrates the flexibility of the DeepAugment approach. It’s possible that systematically optimizing these network parameters could even further improve

	<i>Clean</i>	<i>mCE</i>	Noise			Blur				Weather				Digital			
			Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
ResNet-50	23.9	76.7	80	82	83	75	89	78	80	78	75	66	57	71	85	77	77
+ DeepAugment (EDSR)	23.5	64.0	56	57	54	64	77	71	78	68	64	64	55	64	78	46	67
+ DeepAugment (CAE)	23.2	67.0	58	60	62	62	75	73	77	68	66	60	52	66	80	63	78
+ DeepAugment (Both)	23.3	60.4	49	50	47	59	73	65	76	64	60	58	51	61	76	48	67

Table 1.3: Clean Error, Corruption Error (CE), and mean CE (mCE) values for DeepAugment ablations on ImageNet-C. The mCE value is computed by averaging across all 15 CE values.

performance.

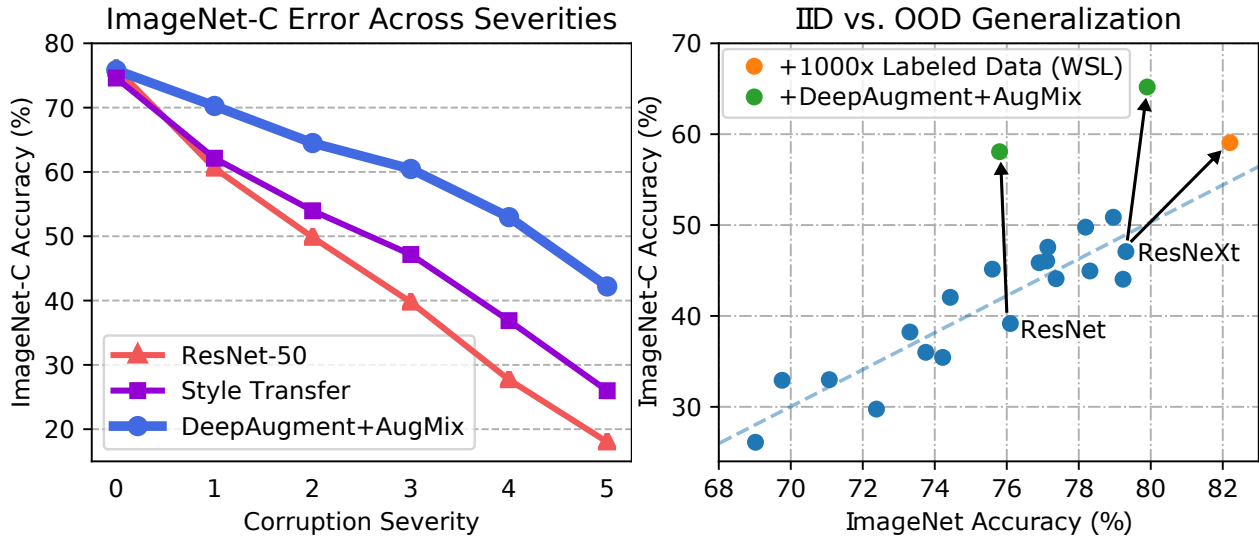


Figure 1.6: The left figure shows accuracy as a function of corruption severity. In the right figure, we show ImageNet accuracy and ImageNet-C accuracy. Previous architectural advances slowly translate to ImageNet-C performance improvements, but the DeepAugment+AugMix robustness intervention on a ResNet-50 approximately yields a 19% accuracy improvement.

1.4.0.3 ImageNet-C.

We now consider a previous robustness benchmark to reassess all four hypotheses. We use the ImageNet-C dataset (Hendrycks and Dietterich [2019]) which applies 15 common image corruptions (e.g., Gaussian noise, defocus blur, simulated fog, JPEG compression, etc.) across 5 severities to ImageNet-1K validation images. We find that DeepAugment improves robustness on ImageNet-C. Figure 1.6 shows that when models are trained with AugMix and DeepAugment, they attain the state-of-the-art, break the trendline, and exceed the corruption robustness provided by training on 1000 \times more labeled training data. Note the augmentations from AugMix and DeepAugment are disjoint from ImageNet-C’s corruptions. This is evidence for the **Larger Models**, **Diverse Data Augmentation**, **Pretraining**, and **Texture Bias** hypotheses.

Chapter 2

Self-Supervised Learning and Model Robustness & Uncertainty

In this chapter, we expand on our goal of trying to improve model robustness by also looking at performance on out-of-distribution (OOD) detection, a measure of model uncertainty, as well as adversarial robustness. In order to improve both model robustness and OOD detection, we turn to Self-Supervised Learning (SSL). We find that self-supervision can benefit robustness in a variety of ways and that self-supervision greatly benefits out-of-distribution detection on difficult, near-distribution outliers - so much so that it exceeds the performance of fully supervised methods.

Again, we focus on Computer Vision as our domain, this time focusing on experiments involving image classification problems on the CIFAR-10 and CIFAR-100 datasets.

2.1 Self-Supervision

High-quality large-scale training datasets for data-hungry tasks such as image classification (e.g. ImageNet) are difficult to create. These training datasets often comprise of millions or more pairs (x_i, y_i) , where x_i is a data point and y_i is a label. Often, the x_i are cheap and easy to acquire. In the case of image classification, one can automate the scraping of millions of images from the internet. However, the labels y_i are often much more expensive to acquire, since they often involve manual human intervention.

Self-supervised learning holds great promise for improving model representations when such labeled data are scarce because SSL manages to extract useful training signals using only the x_i . As a concrete example, given an image dataset $\{x_i\}_{i=0}^N$, one can train a model roughly in the following fashion:

1. For each data point x_i , rotate it by $r \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$ degrees.
2. Feed the rotated image into the model. The model would have a prediction head with 4 logits (one for each possible rotation amount)
3. Backpropagate using r as the true class label.

Intuitively, models must learn something about the structure of images and the classes of images they are shown in order to do well at this task. The internal representations of models trained using SSL in such a fashion can be used for downstream tasks that may otherwise require more data. For example, one can freeze the several layers of the model, replace the head of the model with a small MLP with a softmax output, and train an image classifier using a smaller set of training data $\{(x_i, y_i)\}_{i=0}^{N'}$. There are many different

transformations (apart from rotation) that one can use to transform the training image and construct a true class label with. In the following discussion, however, we only focus on rotations.

To date, self-supervised approaches lag behind fully supervised training on standard accuracy metrics and research has existed in a mode of catching up to supervised performance. Additionally, when used in conjunction with fully supervised learning on a fully labeled dataset, self-supervision has little impact on accuracy. This raises the question of whether large labeled datasets render self-supervision needless.

In the following sections, we show that this is not the case, and that SSL can improve various aspects of model robustness and uncertainty.

2.2 Robustness to Adversarial Perturbations

Improving robustness to adversarial inputs has proven difficult, with adversarial training providing the only longstanding gains (Carlini and Wagner [2017], Athalye, Carlini, and Wagner [2018]). In this section, we demonstrate that auxiliary self-supervision in the form of predicting rotations (Gidaris, Singh, and Komodakis [2018]) can improve upon standard Projected Gradient Descent (PGD) adversarial training (Madry et al. [2018]). We also observe that self-supervision can provide gains when combined with stronger defenses such as TRADES (Zhang et al. [2019]) and is not broken by gradient-free attacks such as SPSA (Uesato et al. [2018]).

2.2.1 Setup

The problem of defending against bounded adversarial perturbations can be formally expressed as finding model parameters θ for the classifier p that minimize the objective

$$\min_{\theta} \mathbb{E}_{(x,y) \sim \mathcal{D}} [\max_{x' \in S} \mathcal{L}_{\text{CE}}(y, p(y | x'); \theta)] \quad \text{where} \quad S = \{x' : \|x - x'\| < \varepsilon\} \quad (2.1)$$

In this paper, we focus on ℓ_{∞} norm bounded adversaries. This means that the norm used to bound S is an ℓ_{∞} norm. (Madry et al. [2018]) propose that PGD is “a universal first-order adversary.” Hence, we first focus on defending against PGD. Let $\text{PGD}(x)$ be the K^{th} step of PGD,

$$x^{k+1} = \Pi_S(x^k + \alpha \text{sign}(\nabla_x \mathcal{L}_{\text{CE}}(y, p(y | x^k); \theta))) \quad \text{and} \quad x^0 = x + U(-\varepsilon, \varepsilon) \quad (2.2)$$

where K is a preset parameter which characterizes the number of steps that are taken, Π_S is the projection operator for the ℓ_{∞} ball S , and $\mathcal{L}_{\text{CE}}(y, p(y | x'); \theta)$ is the loss we want to optimize. Normally, this loss is the cross entropy between the model’s softmax classification output for x and the ground truth label y . For evaluating robust accuracy, we use 20-step and 100-step adversaries. For the 20-step adversary, we set the step-size $\alpha = 2/256$. For the 100-step adversary, we set $\alpha = 0.3/256$ as in Madry et al. [2018]. During training, we use 10-step adversaries with $\alpha = 2/256$.

In all experiments, we use 40-2 Wide Residual Networks (Zagoruyko and Komodakis [2016]). For training, we use SGD with Nesterov momentum of 0.9 and a batch size of 128. We use an initial learning rate of 0.1 and a cosine learning rate schedule (Ilya Loshchilov and Frank Hutter [2016]) and weight decay of 5×10^{-4} . For data augmentation, we use random cropping and mirroring. Hyperparameters were chosen as standard values and are used in subsequent sections unless otherwise specified.

2.2.2 Method

We explore improving representation robustness beyond standard PGD training with auxiliary rotation-based self-supervision (Gidaris, Singh, and Komodakis [2018]). In our approach, we train a classification network along with a separate auxiliary head, which takes the penultimate vector from the network as input and outputs a 4-way softmax distribution. This head is trained along with the rest of the network to predict the

	Clean	20-step PGD	100-step PGD
Normal Training	94.8	0.0	0.0
Adversarial Training	84.2	44.8	44.8
+ Auxiliary Rotations (Ours)	83.5	50.4	50.4

Table 2.1: Results for our defense. All results use $\varepsilon = 8.0/255$. For 20-step adversaries $\alpha = 2.0/255$, and for 100-step adversaries $\alpha = 0.3/255$. More steps do not change results, so the attacks converge. Self-supervision through rotations provides large gains over standard adversarial training.

amount of rotation applied to a given input image (from 0° , 90° , 180° , and 270°). Our overall loss during training can be broken down into a supervised loss and a self-supervised loss

$$\mathcal{L}(x, y; \theta) = \mathcal{L}_{\text{CE}}(y, p(y | \text{PGD}(x)); \theta) + \lambda \mathcal{L}_{\text{SS}}(\text{PGD}(x); \theta). \quad (2.3)$$

Note that the self-supervised component of the loss does not require the ground truth training label y as input. The supervised loss does not make use of our auxiliary head, while the self-supervised loss only makes use of this head. When $\lambda = 0$, our total loss falls back to the loss used for PGD training. For our experiments, we use $\lambda = 0.5$ and the following rotation-based self-supervised loss

$$\mathcal{L}_{\text{SS}}(x; \theta) = \frac{1}{4} \left[\sum_{r \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}} \mathcal{L}_{\text{CE}}(\text{one_hot}(r), p_{\text{rot_head}}(r | R_r(x)); \theta) \right], \quad (2.4)$$

where $R_r(x)$ is a rotation transformation and $\mathcal{L}_{\text{CE}}(r, x; \theta)$ is the cross-entropy between the auxiliary head’s output and the ground-truth label $r \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$. In order to adapt the PGD adversary to the new training setup, we modify the loss used in the PGD update equation (2.2) to maximize both the rotation loss and the classification loss. We report results with the modification here, for completeness. The overall loss that PGD will try to maximize for each training image is $\mathcal{L}_{\text{CE}}(y, p(y | x); \theta) + \mathcal{L}_{\text{SS}}(x; \theta)$. At test-time, the PGD loss does not include the \mathcal{L}_{SS} term, as we want to attack the image classifier and not the rotation classifier.

2.2.3 Results

We are able to attain large improvements over standard PGD training by adding self-supervised rotation prediction. Table 2.1 contains results of our model against PGD adversaries with $K = 20$ and $K = 100$. In both cases, we are able to achieve a 5.6% absolute improvement over classical PGD training. In Figure 2.1 we observe that our method of adding auxiliary rotations actually provides larger gains over standard PGD training as the maximum perturbation distance ε increases. The figure also shows that our method can withstand up to 11% larger perturbations than PGD training without any drop in performance.

In order to demonstrate that our method does not rely on gradient obfuscation, we attempted to attack our models using SPSA Uesato et al. [2018] and failed to notice any performance degradation compared to standard PGD training. In addition, since

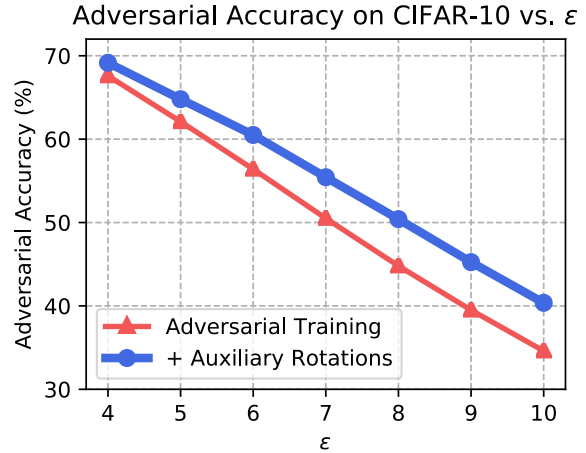


Figure 2.1: The effect of attack strength on a $\varepsilon = 8/255$ adversarially trained model. The attack strengths are $\varepsilon \in \{4/255, 5/255, \dots, 10/255\}$. Since the accuracy gap widens as ε increases, self-supervision’s benefits are masked when observing the clean accuracy alone.

our self-supervised method has the nice property of being easily adaptable to supplement other different supervised defenses, we also studied the effect of adding self-supervised rotations to stronger defenses such as TRADES Zhang et al. [2019]. We found that self-supervision is able to help in this setting as well. Our best-performing TRADES + rotations model gives a 1.22% boost over standard TRADES and a 7.79% boost over standard PGD training in robust accuracy.

2.3 Robustness to Common Corruptions

2.3.1 Setup

In real-world applications of computer vision systems, inputs can be corrupted in various ways that may not have been encountered during training. Improving robustness to these common corruptions is especially important in safety-critical applications. ImageNet-C is a set of fifteen test corruptions and four validation corruptions common corruptions to measure input corruption robustness (Hendrycks and Dietterich [2019]). These corruptions fall into noise, blur, weather, and digital categories. Examples include shot noise, zoom blur, snow, and JPEG compression.

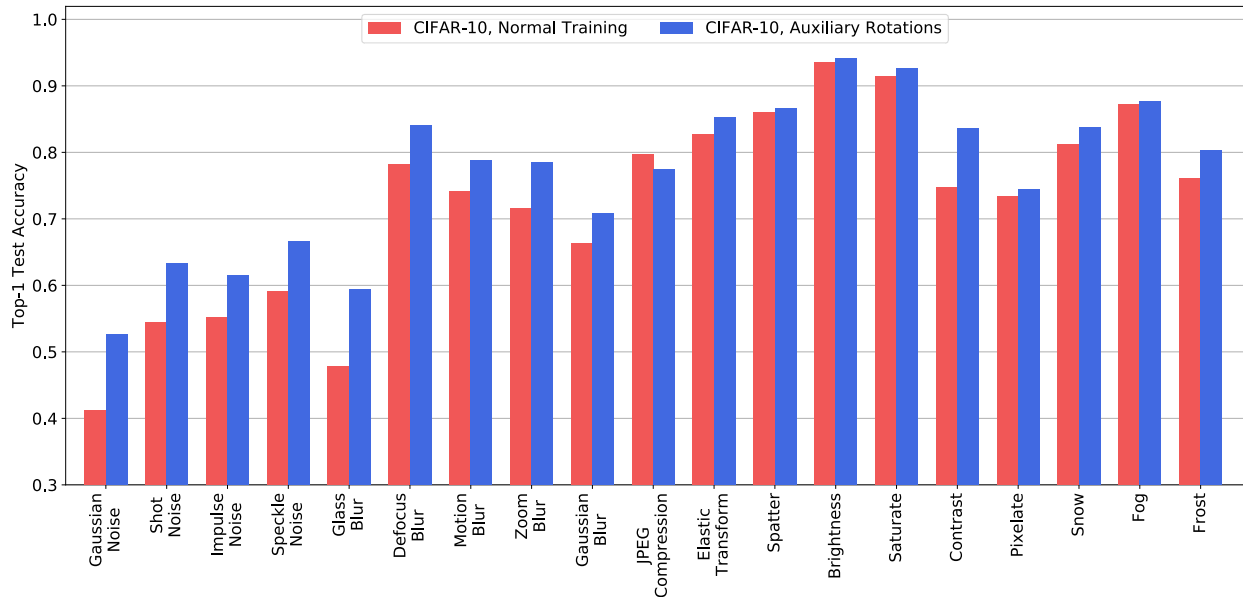


Figure 2.2: A comparison of the accuracy of usual training compared to training with auxiliary rotation self-supervision on the nineteen CIFAR-10-C corruptions. Each bar represents an average over all five corruption strengths for a given corruption type.

We use the CIFAR-10-C validation dataset from the ImageNet-C paper and compare the robustness of normally trained classifiers to classifiers trained with an auxiliary rotation prediction loss. As in the previous section, we predict all four rotations in parallel in each batch. We use 40-2 Wide Residual Networks and the same optimization hyperparameters as before. We do not tune on the validation corruptions, so we report average performance over all corruptions. Results are in Figure 2.2.

2.3.2 Results

The baseline of normal training achieves a clean accuracy of 94.7% and an average accuracy over all corruptions of 72.3%. Training with auxiliary rotations maintains clean accuracy at 95.5% but increases the

average accuracy on corrupted images by 4.6% to 76.9%. Thus, the benefits of self-supervision to robustness are masked by similar accuracy on clean images. Performance gains are spread across corruptions, with a small loss of performance in only one corruption type, JPEG compression. For glass blur, clean accuracy improves by 11.4%, and for Gaussian noise it improves by 11.6%. Performance is also improved by 8.9% on contrast and shot noise and 4.2% on frost, indicating substantial gains in robustness on a wide variety of corruptions. These results demonstrate that self-supervision can regularize networks to be more robust even if clean accuracy is not affected.

2.4 Multi-Class Out-of-Distribution Detection

Self-supervised learning with rotation prediction enables the detection of harder out-of-distribution examples. In this section, we show that self-supervised learning improves out-of-distribution detection when the in-distribution consists in multiple classes.

2.4.1 Setup

In the following experiment, we train a CIFAR-10 classifier and use it as an out-of-distribution detector. When given an example x , we write the classifier’s posterior distribution over the ten classes with $p(y | x)$. Hendrycks and Gimpel [2017] show that $p(y | x)$ can enable the detection of out-of-distribution examples. They show that the maximum softmax probability $\max_c p(y = c | x)$ tends to be higher for in-distribution examples than for out-of-distribution examples across a range of tasks, enabling the detection of OOD examples.

We evaluate each OOD detector using the area under the receiver operating characteristic curve (AUROC) `auroc`. Given an input image, an OOD detector produces an anomaly score. The AUROC is equal to the probability an out-of-distribution example has a higher anomaly score than an in-distribution example. Thus an OOD detector with a 50% AUROC is at random-chance levels, and one with a 100% AUROC is without a performance flaw.

2.4.2 Method

We train a classifier with an auxiliary self-supervised rotation loss. The loss during training is given by:

$$\mathcal{L}_{\text{CE}}(y, p(y | x)) + \left[\sum_{r \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}} \mathcal{L}_{\text{CE}}(\text{one_hot}(r), p_{\text{rot_head}}(r | R_r(x))) \right] \quad (2.5)$$

We only train on in-distribution CIFAR-10 training examples. After training is complete, we score in-distribution CIFAR-10 test set examples and OOD examples with the formula:

$$\text{KL}[U || p(y | x)] + \frac{1}{4} \left[\sum_{r \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}} \mathcal{L}_{\text{CE}}(\text{one_hot}(r), p_{\text{rot_head}}(r | R_r(x))) \right] \quad (2.6)$$

We use the KL divergence of the softmax prediction to the uniform distribution U since it combines well with the rotation score, and because $\text{KL}[U || p(y | x)]$ performs similarly to the maximum softmax probability baseline $\max_c p(y = c | x)$ (Hendrycks, Mazeika, and Dietterich [2019]).

The training loss is standard cross-entropy loss with auxiliary rotation prediction. The detection score is the KL divergence detector from prior work with a rotation score added to it. The rotation score consists of the cross entropy of the rotation softmax distribution to the categorical distribution over rotations with probability 1 at the current rotation and 0 everywhere else. This is equivalent to the negative log probability assigned to the true rotation. Summing the cross entropies over the rotations gives the total rotation score.

\mathcal{D}_{in}	$\mathcal{D}_{\text{out}}^{\text{test}}$	FPR95 ↓		AUROC ↑		AUPR ↑	
		MSP	Rotation	MSP	Rotation	MSP	Rotation
CIFAR-10	Gaussian	8.1	1.2	96.3	99.0	70.8	85.6
	Rademacher	5.9	1.1	97.5	99.1	79.4	86.3
	Blobs	13.3	2.3	94.6	98.9	68.3	86.5
	Textures	45.4	8.9	87.9	97.4	56.2	86.7
	SVHN	25.7	2.7	91.9	98.9	64.0	89.8
	Places365	46.0	38.4	87.7	92.2	57.2	71.3
	LSUN	39.5	28.7	88.5	93.2	57.2	71.0
	CIFAR-100	45.9	44.9	87.2	90.9	54.1	67.7
Mean		28.7	16.0	91.4	96.2	63.4	80.6

Table 2.2: Out-of-distribution example detection results for the maximum softmax probability (MSP) baseline and our rotation method. All results are percentages and the average result of 5 runs.

2.4.3 Results

We evaluate this proposed method against the maximum softmax probability baseline Hendrycks and Gimpel [2017] on a wide variety of anomalies with CIFAR-10 as the in-distribution data. For the anomalies, we select Gaussian, Rademacher, Blobs, Textures, SVHN, Places365, LSUN, and CIFAR-100 images. We observe performance gains across the board and report average AUROC values in [2.3]. On average, the rotation method increases the AUROC by 4.8%.

This method does not require additional data as in Outlier Exposure (Hendrycks, Mazeika, and Dietterich [2019]), although combining the two could yield further benefits. As is, the performance gains are of comparable magnitude to more complex methods proposed in the literature (C. Xie, Wu, et al. [2018]). This demonstrates that self-supervised auxiliary rotation prediction can augment OOD detectors based on fully supervised multi-class representations. More detailed descriptions of the OOD datasets and the full results on each anomaly type with additional metrics are in Table [2.2].

Method	AUROC
Baseline	91.4%
Rotations (Ours)	96.2%

Figure 2.3: OOD detection performance of the maximum softmax probability baseline and our method using self-supervision. Full results are in Chapter ??.

SECOND PART

FORESIGHT

Chapter 3

Mathematical Problem Solving

Many intellectual endeavors require mathematical problem solving, but this skill remains beyond the capabilities of computers.

Although many other tasks have been done very well just by increasing model size (A. Wang et al. [2019]; Zellers et al. [2019]; Huang et al. [2019]; Bisk et al. [2019]; Hendrycks, Basart, et al. [2020]; Hendrycks, Burns, Basart, Zou, et al. [2021]; Hendrycks, Burns, Basart, Critch, et al. [2021]), it is unknown if strong mathematical reasoning can be reached by just scaling models in this manner.

In this section, we develop a new, benchmark to assess true mathematical problem solving and reasoning abilities. Moreover, we find that current state-of-the-art models such as the 1.5 Billion parameter GPT-2 perform poorly on this dataset. If scaling trends continue, our findings indicate that simply increasing budgets and model parameter counts will most likely be an impractical strategy to achieve strong mathematical reasoning.

3.1 The MATH Dataset

To measure this ability in machine learning models, we introduce MATH, a new dataset of 12,500 challenging competition mathematics problems. Each problem in MATH has a full step-by-step solution which can be used to teach models to generate answer derivations and explanations. To facilitate future research and increase accuracy on MATH, we also contribute a large auxiliary pretraining dataset which helps teach models the fundamentals of mathematics. Even though we are able to increase accuracy on MATH, our results show that accuracy remains relatively low, even with enormous Transformer models. Moreover, we find that simply increasing budgets and model parameter counts will be impractical for achieving strong mathematical reasoning if scaling trends continue. While scaling Transformers is automatically solving most other text-based tasks, scaling is not currently solving MATH. To have more traction on mathematical problem solving we will likely need new algorithmic advancements from the broader research community.

While MATH covers advanced problem-solving techniques, models may first need to be trained thoroughly on the fundamentals of mathematics. To address this, we create the first large-scale mathematics pretraining dataset with hundreds of thousands of step-by-step solutions in natural language and \LaTeX . We call this dataset the Auxiliary Mathematics Problems and Solutions (AMPS) pretraining corpus, which consists of Khan Academy and Mathematica data. AMPS has over 100,000 Khan Academy problems with step-by-step solutions in \LaTeX ; these exercises are used to teach human students concepts ranging from basic addition to Stokes' Theorem. It also contains over 5 million problems generated using Mathematica scripts, based on 100 hand-designed modules covering topics such as conic sections, div grad and curl, KL divergence, eigenvalues, polyhedra, and Diophantine equations. In total AMPS contains 23GB of problems and solutions. Domain-specific pretraining Gururangan et al. [2020] on AMPS improves relative accuracy by around 25%, equivalent

Metamath Theorem Proving

To prove: $n \in \mathbb{N} \wedge \frac{n+1}{2} \in \mathbb{N} \implies \exists m \in \mathbb{N} : n = 2m + 1$. GPT-*f*'s generated proof:

```

|- ((N e. NN0 /\ ((N + 1)/2) e. NN0) ->
  ((N - 1) / 2) e. NN0)
|- (N e. NN0 -> N e. CC)
|- 1 e. CC
|- ((N e. CC /\ 1 e. CC) ->
  (N - 1) e. CC ) ....

```

DeepMind Mathematics Dataset

Problem: Divide 1136975704 by -142121963
 Answer: -8
 Problem: Calculate $((-2)/3)/(-1-(-24)/9)$
 Answer: -2/5
 Problem: Let $k(u) = u^2 + u - 4$. Find $k(0)$
 Answer: -4
 Problem: Sort 2, 4, 0, 6
 Answer: 0, 2, 4, 6

MATH Dataset (Ours)

Problem: Tom has a red marble, a green marble, a blue marble, and three identical yellow marbles. How many different groups of two marbles can Tom choose?

Solution: There are two cases here: either Tom chooses two yellow marbles (1 result), or he chooses two marbles of different colors ($\binom{4}{2} = 6$ results). The total number of distinct pairs of marbles Tom can choose is $1 + 6 = \boxed{7}$.

Problem: If $\sum_{n=0}^{\infty} \cos^{2n} \theta = 5$, what is $\cos 2\theta$?

Solution: This geometric series is $1 + \cos^2 \theta + \cos^4 \theta + \dots = \frac{1}{1 - \cos^2 \theta} = 5$. Hence, $\cos^2 \theta = \frac{4}{5}$. Then

$$\cos 2\theta = 2 \cos^2 \theta - 1 = \boxed{\frac{3}{5}}.$$

Problem: The equation $x^2 + 2x = i$ has two complex solutions. Determine the product of their real parts.

Solution: Complete the square by adding 1 to each side. Then $(x + 1)^2 = 1 + i = e^{\frac{i\pi}{4}} \sqrt{2}$, so $x + 1 = \pm e^{\frac{i\pi}{8}} \sqrt[4]{2}$. The desired product is then

$$\left(-1 + \cos\left(\frac{\pi}{8}\right) \sqrt[4]{2}\right) \left(-1 - \cos\left(\frac{\pi}{8}\right) \sqrt[4]{2}\right) = 1 - \cos^2\left(\frac{\pi}{8}\right) \sqrt{2} = 1 - \frac{(1 + \cos(\frac{\pi}{4}))}{2} \sqrt{2} = \boxed{\frac{1 - \sqrt{2}}{2}}.$$

Figure 3.1: Previous work is based on formal theorem provers or straightforward plug-and-chug problems. Our dataset, MATH, has competition mathematics problems with step-by-step solutions written in L^AT_EX and natural language. Models are tasked with generating tokens to construct the final (boxed) answer.

to a 15× increase in model size.

3.1.1 Categorizing Problems.

Problems span various subjects and difficulties. The seven subjects are Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Precalculus. While subjects like Prealgebra are generally easier than Precalculus, within a subject problems can take on different difficulty levels. We

encode a problem’s difficulty level from ‘1’ to ‘5,’ where a subject’s easiest problems for humans are assigned a difficulty level of ‘1,’ and a subject’s hardest problems are assigned a difficulty level of ‘5.’ Concretely, the first few problems of an AMC 8 exam are often level 1, while AIME problems are level 5. This allows us to assess performance across both different subjects and different levels of difficulty.

3.1.2 Formatting

Problems and solutions are consistently formatted using L^AT_EX and the Asymptote vector graphics language. Our usage of L^AT_EX allows us to flexibly encode mathematical problems while avoiding unusual symbols or cumbersome formal languages. Meanwhile, mathematical figures are encoded in the Asymptote language rather than as raster images. This enables pure language models to process figures, diagrams, and graphics, making it possible to assess these models on subjects such as geometry for the first time.

To assess models using exact match, we force the final boxed answers to follow consistent formatting rules. Specifically, probabilities are expressed as simplified fractions. Moreover, matrix entry fractions are encoded with x/y , while all other fractions are consistently encoded with the `\frac{x}{y}` command. Coefficients are encoded without a multiplication symbol (e.g. $5x$ not $5*x$). Expressions with multiple variables are entered in alphabetical order; polynomials are expressed in decreasing degree order. Different fraction encodings equivalent, such as `\frac{x}{y}` and `\dfrac{x}{y}` and x/y . Different parenthesis encodings, such as `\left(` and `(`, are treated as equivalent.

We also allow units to be included or omitted from an answer, we ignore spaces, and we treat common equivalent ways of expressing the same number (e.g., 0.5 and $1/2$, or 0.1 and .1) as the same. When the answer is a factorized polynomial, we permit different orderings of the factors, so that $4(x+1)(x-1)$ is equivalent to $4(x-1)(x+1)$, and so on. These rules cover nearly all ways that different generated or actual solutions can be equivalent in practice.

3.1.3 Automatically Assessing Generated Answers.

Due to design choices in MATH, we can assess the answers generated by a model *automatically*, even though the space of model outputs is combinatorially large. Automatic assessment starts by determining the beginning and end of the answer. This is possible to do even if a model generates step-by-step solutions because the final answers in MATH are wrapped and delimited with the `\boxed{}` command. We can consequently evaluate a model’s output by parsing what is inside the `\boxed{}` command and comparing that with the ground truth answer, while accounting for the equivalent ways of formatting a string described above. Together, the box delimiter and formatting rules provide a unique answer in a well-defined location, which allows us to test for equivalence and use accuracy as our primary metric.

3.1.4 Human-Level Performance.

To estimate human-level performance, we randomly sampled 20 problems from the MATH test set and gave them to humans. We artificially require that the participants have 1 hour to work on the problems and must perform calculations by hand. All participants are university students. One participant who does not like mathematics got $8/20 = 40\%$ correct. A participant ambivalent toward mathematics got $13/20$. Two participants who like mathematics got $14/20$ and $15/20$. A participant who got a perfect score on the AMC 10 exam and attended USAMO several times got $18/20$. A three-time IMO gold medalist got $18/20 = 90\%$, though missed questions were exclusively due to small errors of arithmetic. Expert-level performance is theoretically 100% given enough time, though even 40% accuracy for a machine learning model would be impressive.

Model	Prealgebra	Algebra	Number Theory	Counting & Probability	Geometry	Intermediate Algebra	Precalculus	Average
GPT-2 (0.1B)	5.2	5.1	5.0	2.8	5.7	6.5	7.3	5.4 (+0%)
GPT-2 (0.3B)	6.7	6.6	5.5	3.8	6.9	6.0	7.1	6.2 (+15%)
GPT-2 (0.7B)	6.9	6.1	5.5	5.1	8.2	5.8	7.7	6.4 (+19%)
GPT-2 (1.5B)	8.3	6.2	4.8	5.4	8.7	6.1	8.8	6.9 (+28%)
GPT-3 (2.7B)	2.8	2.9	3.9	3.6	2.1	2.5	2.6	2.9 (−46%)
GPT-3 (175B)	7.7	6.0	4.4	4.7	3.1	4.4	4.0	5.2 (−4%)

Table 3.1: MATH accuracies across subjects for GPT-2 and *few-shot* GPT-3 models. The character ‘B’ denotes the number of parameters in billions. The gray text indicates the *relative* improvement over the 0.1B baseline. All GPT-2 models pretrain on AMPS, and all values are percentages. A 15× increase in model parameters increased accuracy by 1.5%, a 28% relative improvement. Model accuracy is increasing very slowly, so much future research is needed.

3.2 AMPS (Khan + Mathematica) Dataset

Since pretraining data can greatly influence performance Hernandez et al. [2021] and since mathematics is a small fraction of online text, we introduce a large and diverse mathematics pretraining corpus. Our pretraining dataset, the Auxiliary Mathematics Problems and Solutions (AMPS) dataset, has problems and step-by-step solutions typeset in L^AT_EX. AMPS contains over 100,000 problems pulled from Khan Academy and approximately 5 million problems generated from manually designed Mathematica scripts.

3.2.1 Khan Academy.

The Khan Academy subset of AMPS has 693 exercise types with over 100,000 problems and full solutions. Problem types range from elementary mathematics (e.g. addition) to multivariable calculus (e.g. Stokes’ theorem), and are used to teach actual K-12 students. Many of the exercises can be regenerated using code from github.com/Khan/khan-exercises.

3.2.2 Mathematica.

To make AMPS larger, we also contribute our own Mathematica scripts to generate approximately 50× more problems than our Khan Academy dataset. With Mathematica, we designed 100 scripts that test distinct mathematics concepts, 37 of which include full step-by-step L^AT_EX solutions in addition to final answers. We generated around 50,000 exercises from each of our scripts, or around 5 million problems in total. This results in over 23 GB of mathematics problems, making it larger than the 16 GB of natural language used to train BERT Devlin et al. [2019].

Problems include various aspects of algebra, calculus, counting and statistics, geometry, linear algebra, and number theory. Unlike prior approaches to algorithmically generating mathematics problems, we use Mathematica’s computer algebra system so that we can manipulate fractions, transcendental numbers, and analytic functions.

3.3 Experiments

In this section, we perform experiments to investigate performance on the MATH dataset. We find that accuracy remains low even for the best models. Furthermore, unlike for most other text-based datasets, we find that accuracy is increasingly very slowly with model size. If trends continue, then we will need algorithmic improvements, rather than just scale, to make substantial progress on MATH. Nevertheless, we show that making progress is also possible today. We find that pretraining on AMPS increases relative accuracy by 25%, which is comparable to the improvement due to a $15\times$ increase in model size.

We also experiment with using step-by-step solutions. We find that having models generate their own step-by-step solutions before producing an answer actually *degrades* accuracy. We qualitatively assess these generated solutions and find that while many steps remain illogical, they are often related to the question. Finally, we show that step-by-step solutions can still provide benefits today. We find that providing partial ground truth step-by-step solutions can improve performance, and that providing models with step-by-step solutions at training time also increases accuracy.

3.3.1 Setup

Because MATH answers must be generated, we use autoregressive language models, namely GPT-2 (Radford, Metz, and Chintala 2016) and GPT-3 (Brown et al. 2020), which are decoder models pretrained on natural language text. Our GPT-2 models tokenize numbers so that one digit is processed at a time (Henighan et al. 2020). We were unable to get T5 (Raffel et al. 2020), which has a tokenizer that removes many \LaTeX symbols, to have competitive accuracy after a broad hyperparameter sweep.

Before fine-tuning on MATH, models pretrain on AMPS. We pretrain for one epoch, using AdamW (I. Loshchilov and F. Hutter 2019), using a batch size of 128, and using a weight decay of 0.05. We use the standard autoregressive language modeling objective. During pretraining, we upsample Khan Academy data by a factor of 5 and we downsample Mathematica by a factor of 2 to account for the large difference in dataset sizes.

During fine-tuning, models predict final answers and solutions. Concretely, if $\langle P \rangle$ is the problem statement, we train with an equal mix of “ $\langle P \rangle$ Final Answer: $\langle \text{Answer} \rangle$ ” and “ $\langle P \rangle$ Full Solution: $\langle \text{Step-by-Step Solution} \rangle$ ” sequences. This makes it possible for the model to both generate full solutions and also to output just the final answer. For fine-tuning we use the same batch size and weight decay as in pretraining.

Unless otherwise specified, for GPT-2 we use the default HuggingFace (Wolf et al. 2020) generation parameters, except that we use beam search. Our beam search has a beam size of 20 when only generating the final answer, and a beam size of 10 when generating full step-by-step solutions. By default, we evaluate models by prompting them with “ $\langle P \rangle$ Final Answer:” so that they directly generate the final answer to each problem, not the step-by-step solution.

We also evaluate GPT-3 in a few-shot setting (no fine-tuning) using the OpenAI API. We use the ‘Ada’ GPT-3 model which has approximately 2.7 billion parameters, and the ‘Davinci’ model which has approximately 175 billion parameters. Since we are performing few-shot evaluation, we construct our prompt by prepending 8 problems with correct answers (but not step-by-step solutions due to space). Using temperature 0, models output up to 20 tokens for the final answer.

3.3.2 Results

3.3.2.1 Results vs. Model Size

While increasing model parameters often automatically solves many tasks (Brown et al. 2020), we find that MATH is unusually challenging for enormous Transformers. Table 3.1 shows that the average accuracy across subjects for the smallest model, GPT-2 with 0.1 billion parameters, is 5.4%. Meanwhile, a GPT-2 model

with $15\times$ the number of parameters attains 6.9% accuracy, a 28% relative improvement. This indicates that while having more parameters helps, absolute accuracy remains far from the ceiling and is only increasing slowly, quite unlike most other text-based tasks.

3.3.2.2 Results vs. Problem Difficulty

We also analyze model accuracy while controlling for problem difficulty. Higher levels of difficulty correspond to lower accuracy, as expected. These results are visualized in Chapter 3.2. The accuracy of GPT-2 (1.5B) is around 15% for level 1 (easy) and around 4% for level 5 (hard). Even our benchmark’s easiest problems are more challenging than previous benchmarks that focused on straightforward plug-and-chug problems.

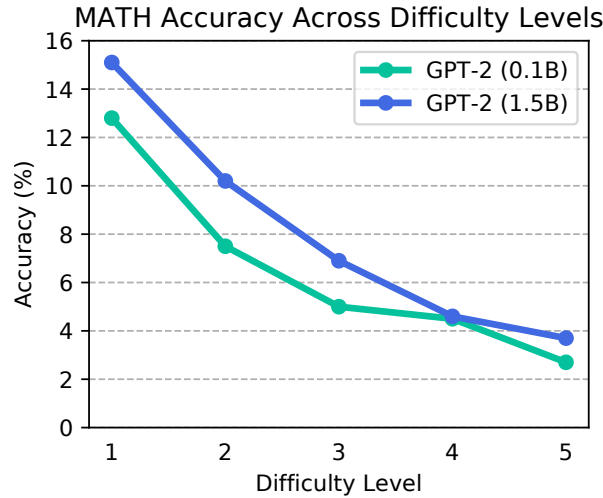


Figure 3.2: Problems that are more difficult for humans are also more difficult for GPT-2.

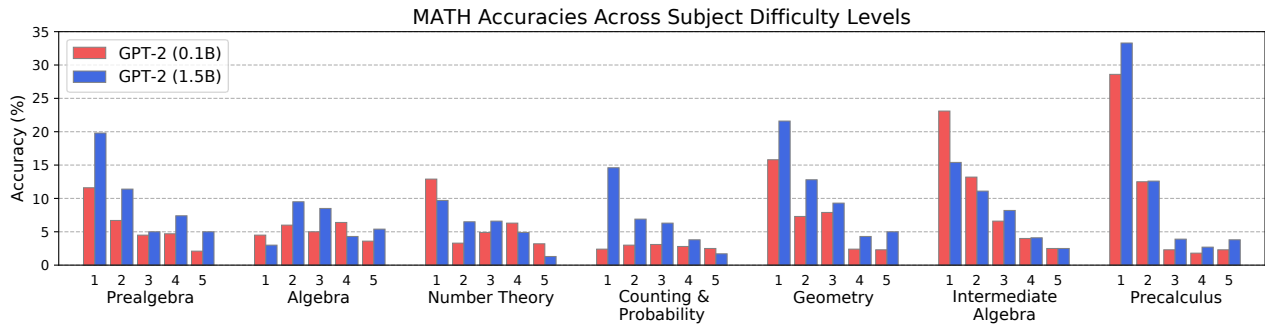
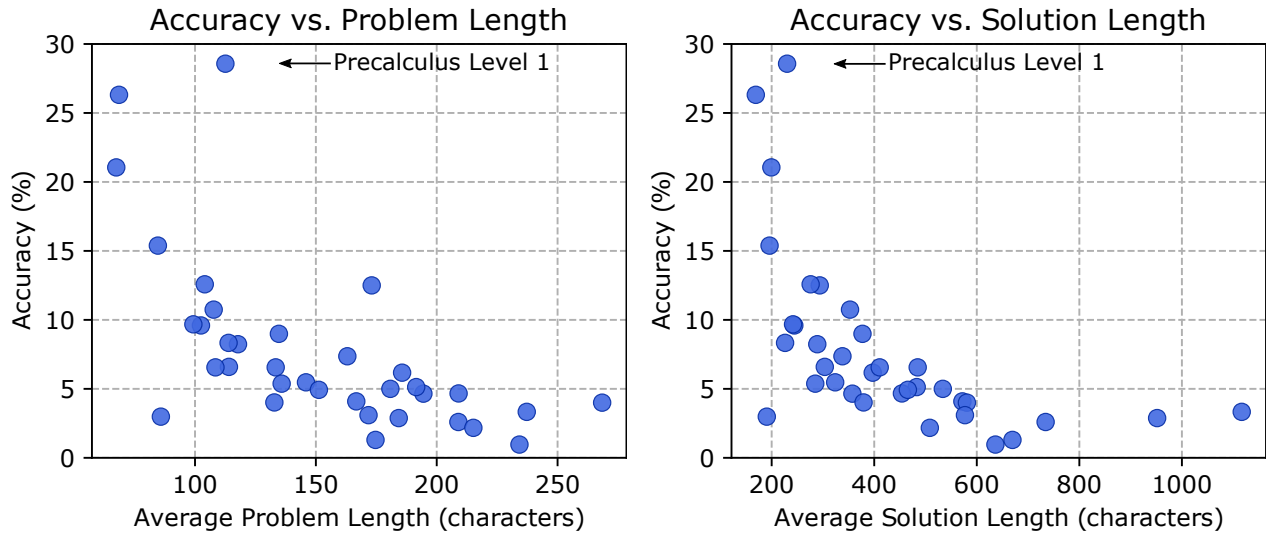


Figure 3.3: Accuracy on MATH per subject per difficulty level, with GPT-2 1.5B and GPT-2 0.1B

3.3.2.3 The effect of AMPS Pretraining

As an ablation, we test how models with AMPS pretraining compare with models that were not pretrained on AMPS. Without pretraining on AMPS, a GPT-2 (1.5B) model fine-tuned on MATH attains 5.5% accuracy. In contrast, a GPT-2 (1.5B) model both pretrained on AMPS and fine-tuned on MATH attains 6.9%, a 25% relative improvement in accuracy. Consequently, AMPS increases accuracy about as much as a $15\times$ increase in parameters, indicating its value as a pretraining dataset.

We tried additionally pretraining on StackExchange, a real-world but less curated source of mathematics text. A GPT-2 (0.3B) model pretrained on both AMPS and questions and answers from Math StackExchange (~ 3



(a) Subject accuracy vs problem length. Each point represents a subject at a specific difficulty level. We exclude problems with asymptote figures. Results are from GPT-2 (1.5B).

(b) Subject accuracy vs solution length. Each point represents a subject at a specific difficulty level. We exclude problems with asymptote figures. Results are from GPT-2 (1.5B).

GB) had 6.0% accuracy, which is actually less than the 6.2% accuracy attained by pretraining on AMPS alone. Thus our dataset is more useful for pretraining even than diverse real-world mathematics data.

3.3.2.4 Error Detection

To determine whether we can trust the answers from a model, we analyze model confidence to see whether confidence tends to be higher for correct answers. We define confidence as the average prediction probability of the tokens that make up a generated answer. We histogram confidences for correct and incorrect answers in Figure 3.5. GPT-2 (1.5B) is highly overconfident, with confidences that are typically around 100%, and there is substantial overlap between correct and incorrect answers.

Following Hendrycks and Gimpel [2017], we computed the probability that a correct answer has higher confidence than an incorrect answer. To do this, we compute the Area Under the Receiver Operating Characteristic curve (AUROC). An AUROC of 100% corresponds to being able to perfectly detect correct and incorrect answers, while 50% corresponds to random chance. We find that with GPT-2 (1.5B), the AUROC is quite low at 68.8%. This suggests there is substantial room for improvement in detecting model errors.

3.3.2.5 The Benefits of MATH Solutions

We find that giving models partial step-by-step MATH solutions during inference can improve accuracy. We test performance when we allow models to predict the final answer given a “hint” in the form of a portion of the ground truth step-by-step solution. To do so, for this experiment we prompt models with “(P) <Partial Step-by-Step Solution without Final Answer> Final Answer:” during both fine-tuning and evaluation for different partial fractions of the step-by-step solution. This is the same as the default setting when we let models see 0% of the step-by-step solution. When models see “99%” of the solution, they are given the whole step-by-step solution except for the final answer. We show results with GPT-2 (0.7B) for different fractions of the solution in Figure 3.6. Observe that the model still only attains approximately 40% when given 99% of the solution, indicating room for improvement.

Finally, we also find that providing models with step-by-step during training can further improve performance. We run an ablation by fine-tuning models on MATH with the same setup as before, except that we only show

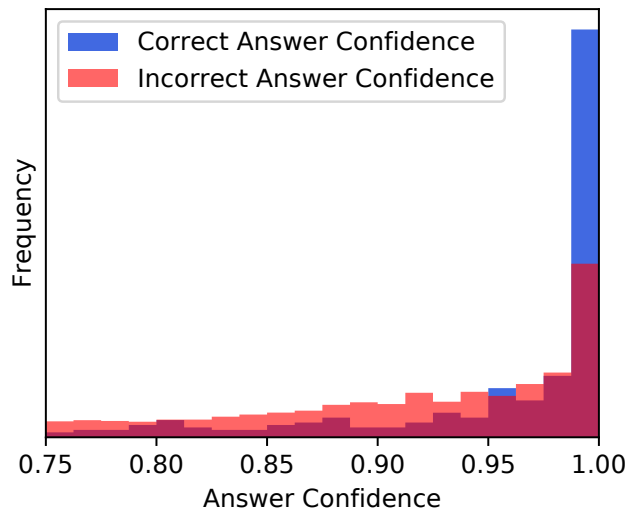


Figure 3.5: Histogram of the answer confidences of GPT-2 1.5B. The incorrect answer histogram is translucent and overlays the correct answer histogram. GPT-2 1.5B is overconfident and has an AUROC of 68.8%, indicating that it is not good at detecting errors from confidence alone. Confidences below 0.75 are omitted for ease of visualization.

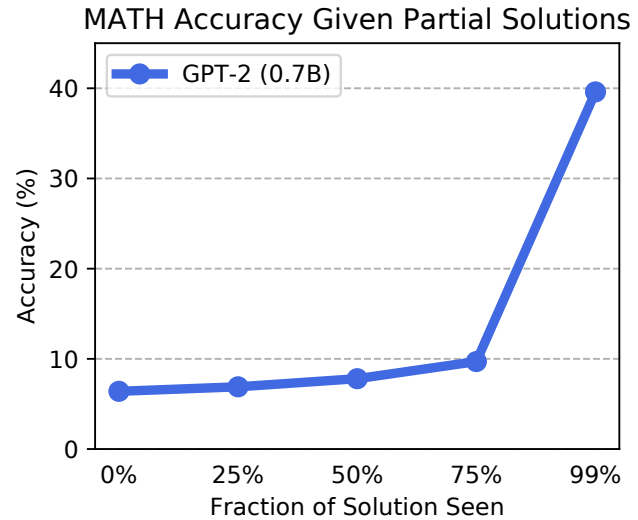


Figure 3.6: Models conditioned on most of a problem’s step-by-step solution can often understand the solution to predict the final answer. ‘99%’ of a solution is all the solution text before the final answer. Not all solutions have an answer that is immediate from from the preceding solution text.

examples with the final answer and no step-by-step solution. If we fine-tune with only the final answer, the GPT-2 (1.5B) accuracy decreases by 0.6% to 6.3%, a 10% relative reduction.

Conclusion

In Part 1, we made progress in improving model robustness by using new data augmentation techniques and self-supervision. We also introduced a new dataset (ImageNet-R) that helped us benchmark our new data augmentation strategy (DeepAugment).

In our analysis of DeepAugment in Chapter 1, we saw that even the best training setup for our ResNet-50s (DeepAugment + AugMix) still had a 53.2% error rate on ImageNet-R (45.2% higher than the error rate on ImageNet-200). It is promising that DeepAugment is the first neural-network based data augmentation strategy, and still outperforms classic strategies. In this vein, Noise2Net demonstrated the flexibility of the DeepAugment approach. It's possible that instead of randomly initializing network parameters, systematically optimizing these parameters could even further improve performance. This is an avenue of research we leave to future work. In fact, at the time of this writing, work has already built upon DeepAugment and shown that this is true (Calian et al. [2021](#)).

We also introduced self-supervised learning as a possible avenue of research to improve model robustness. Whereas in Chapter 1, where we mainly looked at model robustness through the lens of image corruptions and ImageNet-R, in Chapter 2, we expanded our analysis to include out-of-distribution detection and adversarial robustness, as well as robustness to common corruptions. Surprisingly, we showed that self-supervised learning provides benefits on all of these axes, making it a promising avenue for future robustness research. Specifically, we leave experiments such as testing out new self-supervised objectives (instead of rotation prediction) and new loss integration techniques (instead of our additive self-supervised loss) to future work.

In Part 2, we took steps to understand how we might best cope with AI in the future. Our core question was to evaluate the reasoning capabilities of large language models. It is critical to have some foresight regarding model performance on difficult reasoning tasks to guide the decisions we make on how to best handle a future with such powerful AI. We introduced the MATH, a very difficult benchmark and dataset for evaluating automated reasoning on mathematics problems. We showed that the research community has a long way to go in terms of creating models that can reason as well as humans - a good thing, since it gives us as a society more time to decide on steps forward to prepare for increasingly powerful AI. We hope that MATH will help in keeping track of reasoning capabilities over the coming years.

We only mentioned mathematical problem solving as a way to measure the reasoning ability of large language models. Several benchmarks that measure reasoning capabilities will always be more useful than a singular benchmark. A future avenue of research is to expand on this with new tasks that require reasoning. One promising direction is programming challenge problem solving: giving models natural language text defining a programming challenge as input, and training them to generate code to solve the problem. Such continued work would help us forecast and prepare for AGI.

Bibliography

- Athalye, Anish, Nicholas Carlini, and David Wagner (July 2018). “Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples”. In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*.
- Biederman, Irving and Ginny Ju (1988). “Surface versus edge-based determinants of visual recognition”. In: *Cognitive psychology* 20.1, pp. 38–64.
- Bisk, Yonatan et al. (2019). *PIQA: Reasoning about Physical Commonsense in Natural Language*.
- Brown, T. et al. (2020). “Language Models are Few-Shot Learners”. In: *ArXiv* abs/2005.14165.
- Calian, Dan A. et al. (2021). *Defending Against Image Corruptions Through Adversarial Augmentations*. arXiv: [2104.01086 \[cs.CV\]](#).
- Carlini, Nicholas and David Wagner (2017). *Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods*.
- Deng, Jia (2012). *Large scale visual recognition*. Tech. rep. PRINCETON UNIV NJ DEPT OF COMPUTER SCIENCE.
- Devlin, J. et al. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *NAACL-HLT*.
- Dodge, Samuel and Lina Karam (2017). “A study and comparison of human and deep learning recognition performance under visual distortions”. In: *2017 26th international conference on computer communication and networks (ICCCN)*. IEEE, pp. 1–7.
- Gao, Shanghua et al. (2019). “Res2Net: A New Multi-scale Backbone Architecture”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. ISSN: 1939-3539. DOI: [10.1109/tpami.2019.2938758](#). URL: <http://dx.doi.org/10.1109/TPAMI.2019.2938758>.
- Geirhos, Robert, Jörn-Henrik Jacobsen, et al. (2020). “Shortcut Learning in Deep Neural Networks”. In: *arXiv preprint arXiv:2004.07780*.
- Geirhos, Robert, Patricia Rubisch, et al. (2019). “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness”. In: *ICLR*.
- Gidaris, Spyros, Praveer Singh, and Nikos Komodakis (2018). “Unsupervised Representation Learning by Predicting Image Rotations”. In: *International Conference on Learning Representations*.
- Gururangan, Suchin et al. (2020). “Don’t Stop Pretraining: Adapt Language Models to Domains and Tasks”. In: *ArXiv* abs/2004.10964.
- He, Kaiming et al. (2015). *Deep residual learning for image recognition*. *CoRR* abs/1512.03385 (2015).
- Hendrycks, Dan, Steven Basart, et al. (2020). *The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization*. arXiv: [2006.16241 \[cs.CV\]](#).
- Hendrycks, Dan, Collin Burns, Steven Basart, Andrew Critch, et al. (2021). “Aligning AI With Shared Human Values”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Hendrycks, Dan, Collin Burns, Steven Basart, Andy Zou, et al. (2021). “Measuring Massive Multitask Language Understanding”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*.
- Hendrycks, Dan and Thomas Dietterich (2019). “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations”. In: *ICLR*.

- Hendrycks, Dan and Kevin Gimpel (2017). “A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks”. In: *ICLR*.
- Hendrycks, Dan, Kimin Lee, and Mantas Mazeika (2019). “Using Pre-Training Can Improve Model Robustness and Uncertainty”. In: *ICML*.
- Hendrycks, Dan, Xiaoyuan Liu, et al. (2020). “Pretrained Transformers Improve Out-of-Distribution Robustness”. In: *ACL*.
- Hendrycks, Dan, Mantas Mazeika, and Thomas Dietterich (2019). “Deep Anomaly Detection with Outlier Exposure”. In: *International Conference on Learning Representations*.
- Hendrycks, Dan, Norman Mu, et al. (2020). “AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty”. In: *ICLR*.
- Hendrycks, Dan, Kevin Zhao, et al. (2019). “Natural Adversarial Examples”. In: *ArXiv* abs/1907.07174.
- Henighan, T. et al. (2020). “Scaling Laws for Autoregressive Generative Modeling”. In: *ArXiv* abs/2010.14701.
- Hernandez, Danny et al. (2021). “Scaling Laws for Transfer”. In: *arXiv*.
- Huang, Lifu et al. (2019). *Cosmos QA: Machine Reading Comprehension with Contextual Commonsense Reasoning*.
- Itakura, Shoji (July 1994). “Recognition of Line-Drawing Representations by a Chimpanzee (Pan troglodytes)”. In: *The Journal of General Psychology* 121.3, pp. 189–197. DOI: [10.1080/00221309.1994.9921195](https://doi.org/10.1080/00221309.1994.9921195).
- Kolesnikov, Alexander et al. (2019). “Large Scale Learning of General Visual Representations for Transfer”. In: *arXiv preprint arXiv:1912.11370*.
- Lim, Bee et al. (2017). “Enhanced deep residual networks for single image super-resolution”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 136–144.
- Loshchilov, I. and F. Hutter (2019). “Decoupled Weight Decay Regularization”. In: *ICLR*.
- Loshchilov, Ilya and Frank Hutter (2016). “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: *ICLR*.
- Madry, Aleksander et al. (2018). “Towards Deep Learning Models Resistant to Adversarial Attacks”. In: *ICLR*.
- Mahajan, Dhruv et al. (2018). “Exploring the Limits of Weakly Supervised Pretraining”. In: *ECCV*.
- Mordvintsev, Alexander, Christopher Olah, and Mike Tyka (2015). “Inceptionism: Going deeper into neural networks”. In: *arXiv*.
- Orhan, A. Emin (2019). “Robustness properties of Facebook’s ResNeXt WSL models”. In: *ArXiv* abs/1907.07640.
- Paul Christiano: *Current Work in AI Alignment* (Apr. 2020). URL: <https://www.effectivealtruism.org/articles/paul-christiano-current-work-in-ai-alignment/>.
- Radford, A., Luke Metz, and Soumith Chintala (2016). “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *CoRR* abs/1511.06434.
- Raffel, Colin et al. (2020). “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *JMLR*.
- Rusak, Evgenia et al. (2020). “Increasing the robustness of DNNs against image corruptions by playing the Game of Noise”. In: *arXiv preprint arXiv:2001.06057*.
- Tanaka, Masayuki (Dec. 2006). “Recognition of pictorial representations by chimpanzees (Pan troglodytes)”. In: *Animal Cognition* 10.2, pp. 169–179. DOI: [10.1007/s10071-006-0056-1](https://doi.org/10.1007/s10071-006-0056-1).
- Taori, Rohan et al. (2020). *When Robustness Doesn’t Promote Robustness: Synthetic vs. Natural Distribution Shifts on ImageNet*. URL: <https://openreview.net/forum?id=HyxPIyrFvH>.
- Theis, Lucas et al. (2017). “Lossy image compression with compressive autoencoders”. In: *arXiv preprint arXiv:1703.00395*.
- Uesato, Jonathan et al. (2018). “Adversarial risk and the dangers of evaluating against weak attacks”. In: *arXiv preprint arXiv:1802.05666*.
- Wang, Alex et al. (2019). “SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems”. In: *NeurIPS*.
- Wang, Haohan et al. (2019). *Learning Robust Global Representations by Penalizing Local Predictive Power*.
- Wolf, Thomas et al. (2020). “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics.

- Wong, Eric, Leslie Rice, and J Zico Kolter (2020). “Fast is better than free: Revisiting adversarial training”. In: *arXiv preprint arXiv:2001.03994*.
- Xie, Cihang, Yuxin Wu, et al. (2018). “Feature Denoising for Improving Adversarial Robustness”. In: *arXiv preprint*.
- Xie, Cihang and Alan Yuille (2020). “Intriguing Properties of Adversarial Training at Scale”. In: *International Conference on Learning Representations*.
- Xie, Saining et al. (2016). “Aggregated residual transformations for deep neural networks. 2016”. In: *arXiv preprint arXiv:1611.05431*.
- Yin, Dong et al. (2019). “A Fourier perspective on model robustness in computer vision”. In: *arXiv preprint arXiv:1906.08988*.
- Zagoruyko, Sergey and Nikos Komodakis (2016). “Wide Residual Networks”. In: *BMVC*.
- Zellers, Rowan et al. (2019). *HellaSwag: Can a Machine Really Finish Your Sentence?* arXiv: [1905.07830](#) [\[cs.CL\]](#).
- Zhang, Hongyang et al. (2019). “Theoretically Principled Trade-off between Robustness and Accuracy”. In: *arXiv preprint arXiv:1901.08573*.