PIXMIX: Dreamlike Pictures Comprehensively Improve Safety Measures

Dan Hendrycks*
UC Berkeley

Andy Zou*
UC Berkeley

Mantas Mazeika
UIUC

Leonard Tang
Harvard University

Bo Li
UIUC

Dawn Song
UC Berkeley

Jacob Steinhardt
UC Berkeley

2022 CVPR

Presenter : Jason Lee

About PIXMIX?

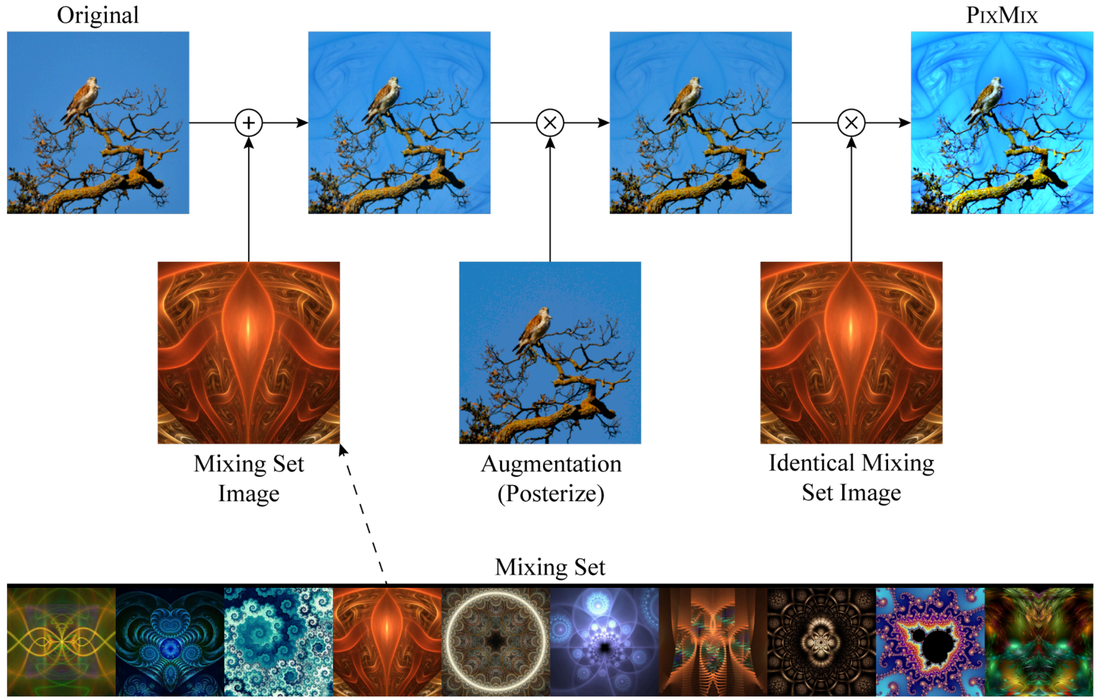


Figure 2. Top: An instance of a PIXMIX augmentation being applied to a bird image. The original clean image is mixed with augmented versions of itself and an image such as a fractal. Bottom: Sample images from the PIXMIX mixing set. We select fractals and feature visualizations from manually curated online sources. In ablations, we find that these new sources of visual structure for augmentations outperform numerous synthetic image distributions explored in prior work [2].

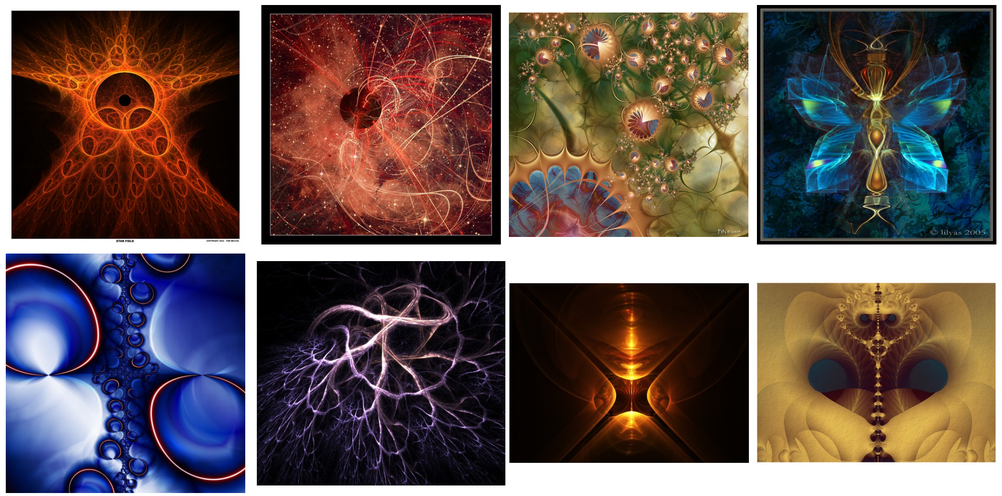
```
def pixmix(x_orig, x_mixing_pic, k=4, beta=3):
    x_pixmix = random.choice([augment(x_orig), x_orig])

    for i in range(random.choice([0,1,...,k])): # random count of mixing rounds

        # mixing_pic is from the mixing set (e.g., fractal, natural image, etc.)
        mix_image = random.choice([augment(x_orig), x_mixing_pic])
        mix_op = random.choice([additive, multiplicative])

        x_pixmix = mix_op(x_pixmix, mix_image, beta)

    return x_pixmix

def augment(x):
    aug_op = random.choice([rotate, solarize, ..., posterize])
    return aug_op(x)
```

Figure 3. Simplified code for PIXMIX, our proposed data augmentation method. Initial images are mixed with a randomly selected image from our mixing set or augmentations of the clean image. The mixing operations are selected at random, and the mixing set includes fractals and feature visualization pictures. PIXMIX integrates new complex structures into the training process by leveraging fractals and feature visualizations, resulting in improved classifier robustness and uncertainty estimation across numerous safety measures.

Pix : a set of structurally complex pictures

Mix : pipeline for augmenting clean training pictures

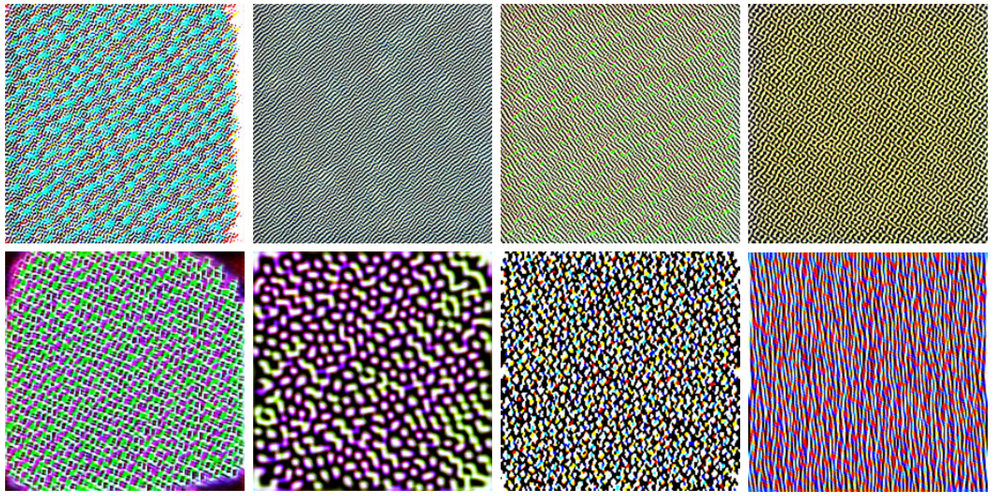
PIX



Fractals

(14,230 images from DevianArt)

+



Feature Visualization

(4,700 images from OpenAI Microscope)

“Dreamlike Pictures”

MIX

```
def pixmix(x_orig, x_mixing_pic, k=4, beta=3):
    x_pixmix = random.choice([augment(x_orig), x_orig])

    for i in range(random.choice([0,1,...,k])): # random count of mixing rounds

        # mixing_pic is from the mixing set (e.g., fractal, natural image, etc.)
        mix_image = random.choice([augment(x_orig), x_mixing_pic])
        mix_op = random.choice([additive, multiplicative])

        x_pixmix = mix_op(x_pixmix, mix_image, beta)

    return x_pixmix

def augment(x):
    aug_op = random.choice([rotate, solarize, ..., posterize])
    return aug_op(x)
```

```python
def pixmix(orig, mixing_pic, preprocess):

  mixings = utils.mixings
  tensorize, normalize = preprocess['tensorize'], preprocess['normalize']
  if np.random.random() < 0.5:
    mixed = tensorize(augment_input(orig))
  else:
    mixed = tensorize(orig)

  for _ in range(np.random.randint(args.k + 1)):

    if np.random.random() < 0.5:
      aug_image_copy = tensorize(augment_input(orig))
    else:
      aug_image_copy = tensorize(mixing_pic)

    mixed_op = np.random.choice(mixings)
    mixed = mixed_op(mixed, aug_image_copy, args.beta)
    mixed = torch.clip(mixed, 0, 1)

  return normalize(mixed)
```

```python

augmentations = [
    autocontrast, equalize, posterize, rotate, solarize, shear_x, shear_y,
    translate_x, translate_y
]

augmentations_all = [
    autocontrast, equalize, posterize, rotate, solarize, shear_x, shear_y,
    translate_x, translate_y, color, contrast, brightness, sharpness
]

######################################################
###################### MIXINGS #######################
######################################################

def get_ab(beta):
  if np.random.random() < 0.5:
    a = np.float32(np.random.beta(beta, 1))
    b = np.float32(np.random.beta(1, beta))
  else:
    a = 1 + np.float32(np.random.beta(1, beta))
    b = -np.float32(np.random.beta(1, beta))
  return a, b

def add(img1, img2, beta):
  a,b = get_ab(beta)
  img1, img2 = img1 * 2 - 1, img2 * 2 - 1
  out = a * img1 + b * img2
  return (out + 1) / 2

def multiply(img1, img2, beta):
  a,b = get_ab(beta)
  img1, img2 = img1 * 2, img2 * 2
  out = (img1 ** a) * (img2.clip(1e-37) ** b)
  return out / 2

mixings = [add, multiply]

```

PIXMIX is good for ML Safety

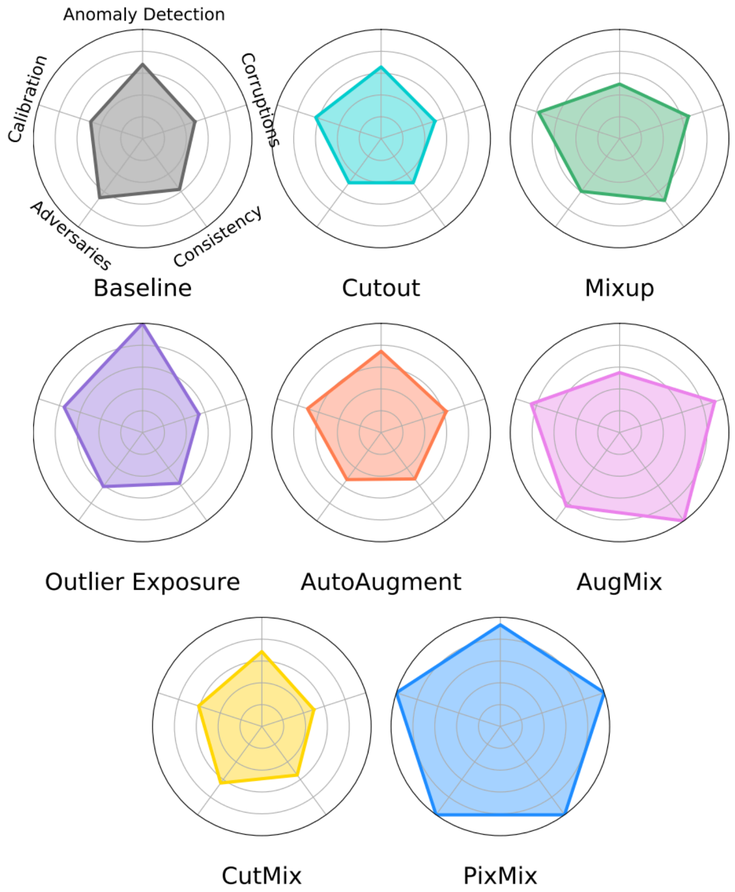


Figure 1. Normalized performance of different methods on five different model safety measures. PIXMIX is the only method that significantly outperforms the baseline in all five safety measures.

- Corruptions : classify corrupted images
- Consistency : consistently classify perturbed images
- Adversaries : classify images that have been adversarially perturbed by projected gradient descent
- Calibration : classify images with calibrated prediction probabilities
- Anomaly Detection : detect out-of-distribution or out-of-class images from various unseen distributions

Dataset & Data Augmentations

Figure 4. We comprehensively evaluate models across safety tasks, including corruption robustness (ImageNet-C, ImageNet-$\overline{\text{C}}$), rendition robustness (ImageNet-R), prediction consistency (ImageNet-P), confidence calibration, and anomaly detection. ImageNet-C [15] contains 15 common corruptions, including fog, snow, and motion blur. ImageNet-$\overline{\text{C}}$ [33] contains additional corruptions. ImageNet-R [13] contains renditions of object categories and measures robustness to shape abstractions. ImageNet-P [15] contains sequences of gradual perturbations to images, across which predictions should be consistent. Anomalies are semantically distinct from the training classes. Existing work focuses on learning representations that improve performance on one or two metrics, often to the detriment of others. Developing models that perform well across multiple safety metrics is an important next step.

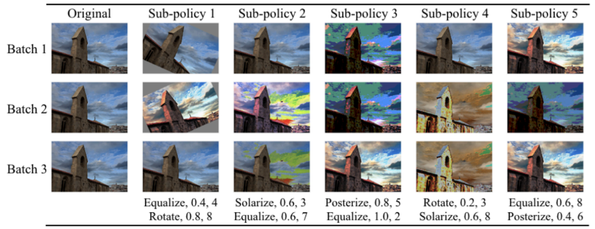


Figure 3. One of the successful policies on ImageNet. As described in the text, most of the policies found on ImageNet used color-based transformations.

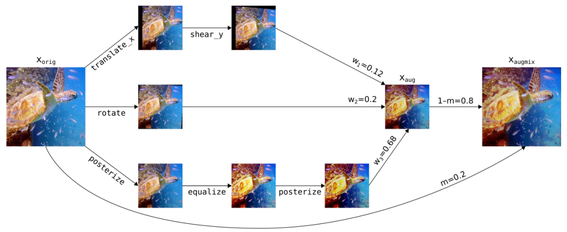


Figure 4: A realization of AUGMIX. Augmentation operations such as translate_x and weights such as m are randomly sampled. Randomly sampled operations and their compositions allow us to explore the semantically equivalent input space around an image. Mixing these images together produces a new image without veering too far from the original.

Experiments - CIFAR-10, CIFAR-100

	Accuracy	Corruptions		Consistency		Adversaries	Calibration			Anomaly	
	Clean	C	$\overline{\text{C}}$	CIFAR-P		PGD	Clean	C	$\overline{\text{C}}$	Detection	
	Error	mCE	mCE	mFR	mT5D	Error	RMS	RMS	RMS	AUROC (↑)	AUPR (↑)
Baseline	21.3	50.0	52.0	10.7	2.7	96.8	14.6	31.2	30.9	77.7	35.4
Cutout	19.9	51.5	50.2	11.9	2.7	98.5	11.4	31.1	29.4	74.3	31.3
Mixup	21.1	48.0	49.8	9.5	3.0	97.4	10.5	13.0	12.9	71.7	31.9
CutMix	20.3	51.5	49.6	12.0	3.0	97.0	12.2	29.3	26.5	74.4	32.3
AutoAugment	**19.6**	47.0	46.8	11.2	2.6	98.1	9.9	24.9	22.8	80.4	33.2
AugMix	20.6	35.4	41.2	6.5	1.9	95.6	12.5	18.8	22.5	84.9	53.8
OE	21.9	50.3	52.1	11.3	3.0	97.0	12.0	13.8	13.9	**90.3**	66.2
PIXMIX	20.3	**30.5**	**36.7**	**5.7**	**1.6**	**92.9**	**7.0**	**8.1**	**8.9**	89.3	**70.9**

Table 7. Full results for CIFAR-100. mT5D is an additional metric used for gauging prediction consistency in ImageNet-P, which we adapt to CIFAR-100. Note PIXMIX can achieve 19.6% error rate if it uses 300K Random Images as the Mixing Set, so PIXMIX can achieve the same accuracy as AutoAugment yet also do better on safety metrics.

	Accuracy	Corruptions		Consistency		Adversaries	Calibration			Anomaly	
	Clean	CIFAR-C	$\overline{\text{C}}$	CIFAR-P		PGD	Clean	CIFAR-C	$\overline{\text{C}}$	Detection	
	Error	mCE	mCE	mFR	mT5D	Error	RMS	RMS	RMS	AUROC (↑)	AUPR (↑)
Baseline	4.4	26.4	26.4	3.4	1.7	91.3	6.4	22.7	22.4	91.9	70.9
Cutout	**3.6**	25.9	24.5	3.7	1.7	96.0	3.3	17.8	17.5	91.4	63.6
Mixup	4.2	21.0	22.1	2.9	2.1	93.3	12.5	12.1	10.9	88.2	67.1
CutMix	4.0	26.5	25.4	3.5	2.1	92.1	5.0	18.6	17.8	92.0	65.5
AutoAugment	3.9	22.2	24.4	3.6	1.7	95.1	4.0	14.8	16.6	93.2	64.6
AugMix	4.3	12.4	16.4	1.7	1.2	86.8	5.1	9.4	12.6	89.2	61.5
OE	4.6	25.1	26.1	3.4	1.9	92.9	6.9	13.0	13.2	**98.4**	**92.5**
PIXMIX	4.2	**9.5**	**13.6**	**1.7**	**1.0**	**82.1**	**2.6**	**3.7**	**5.3**	97.0	88.4

Table 8. Full results for CIFAR-10. mT5D is an additional metric used for gauging prediction consistency in ImageNet-P, which we adapt to CIFAR-10.

- Model : 40-4 Wide ResNet
- FIXMIX : k = 4, beta = 3
- PGD : 2/255 budget, 20 steps

Experiments - ImageNet

	Accuracy	Robustness			Consistency		Calibration				Anomaly	
	Clean	C	$\overline{\text{C}}$	R	ImageNet-P		Clean	C	$\overline{\text{C}}$	R	Detection	
	Error	mCE	Error	Error	mFR	mT5D	RMS	RMS	RMS	RMS	AUROC (↑)	AUPR (↑)
Baseline	23.9	78.2	61.0	63.8	58.0	78.4	5.6	12.0	20.7	19.7	79.7	48.6
Cutout	22.6	76.9	60.2	64.8	57.9	75.2	3.8	11.1	17.1	14.6	81.7	49.6
Mixup	22.7	72.7	55.0	62.3	54.3	73.2	5.8	7.3	13.2	44.6	72.2	51.3
CutMix	22.9	77.8	59.8	66.5	60.3	76.6	6.2	9.1	15.3	43.5	78.4	47.9
AutoAugment	**22.4**	73.8	58.0	61.9	54.2	72.0	**3.6**	8.0	14.3	12.6	84.4	58.2
AugMix	22.8	71.0	56.5	61.7	52.7	70.9	4.5	9.2	15.0	13.2	84.2	61.1
SIN	25.4	70.9	57.6	**58.5**	54.4	71.8	4.2	6.5	14.0	16.2	84.8	62.3
PIXMIX	22.6	**65.8**	**44.3**	60.1	**51.1**	**69.1**	**3.6**	**6.3**	**5.8**	**11.0**	**85.7**	**64.1**

Table 9. Full results for ImageNet. mT5D is an additional metric used for gauging prediction consistency in ImageNet-P. **Bold** is best, and underline is second best.

	Accuracy	Robustness			Consistency		Calibration				Anomaly	
	Clean	C	$\overline{\text{C}}$	R	ImageNet-P		Clean	C	$\overline{\text{C}}$	R	Detection	
	Error	mCE	Error	Error	mFR	mT5D	RMS	RMS	RMS	RMS	AUROC (↑)	AUPR (↑)
Baseline	23.9	78.2	61.0	63.8	58.0	78.4	5.6	12.0	20.7	19.7	79.7	48.6
ANT	23.9	67.0	61.0	61.0	48.0	68.4	7.0	10.3	19.3	22.9	80.9	54.3
Speckle	24.2	72.7	62.1	62.1	51.2	70.6	5.6	11.6	19.8	20.9	79.7	53.3
Noise2Net	22.7	71.6	57.7	57.6	51.5	72.3	4.4	8.9	16.3	15.2	84.8	60.4
EDSR	23.5	65.4	54.7	60.3	44.6	63.3	4.5	8.4	15.7	16.7	71.7	36.3
ℓ_∞ Adversarial	45.5	92.6	68.0	65.2	38.5	41.5	15.5	10.2	15.1	10.2	69.8	26.4
ℓ_2 Adversarial	37.2	85.5	64.9	63.0	29.2	34.8	11.3	9.7	16.6	10.7	78.9	40.2

Table 11. While many noise-based augmentation methods often do well on ImageNet-C by targeting the noise corruptions, they do not reliably improve performance across many safety metrics.

- Model : pre-trained ResNet-50
- FIXMIX : k = 4, beta = 4
- Finetune for 90 epochs
- No adversaries results cause all tested methods accuracy declined to zero

- Since noise-based augmentations sometimes nearly overlap with the test distribution and thereby may have an unfair advantage, separately compare

Ablations - Choice of Mixing Set

		Accuracy	Corruptions	Consistency	Adversaries	Calibration	Anomaly
	PIXMIX Mixing Set	Clean Error	C mCE	CIFAR-P mFR	PGD Error	C RMS	Detection AUROC (↑)
Previous	Dead Leaves (Squares) [2]	21.3	36.2	6.3	94.1	15.8	81.8
	Spectrum + Color + WMM [2]	20.7	36.1	6.6	94.4	15.9	85.8
	StyleGAN (Oriented) [2]	20.4	37.3	7.2	97.0	14.9	83.7
	FractalDB [22]	<u>20.3</u>	33.9	6.4	98.2	12.0	82.5
	300K Random Images [19]	**19.6**	34.5	6.3	94.7	12.9	86.2
New	Fractals	<u>20.3</u>	32.3	6.2	95.5	<u>8.7</u>	<u>88.9</u>
	Feature Visualization (FVis)	21.5	**30.3**	**5.4**	**91.5**	9.9	88.1
	Fractals + FVis	<u>20.3</u>	<u>30.5</u>	<u>5.7</u>	<u>92.9</u>	**8.1**	**89.3**

Table 4. Mixing set ablations showing that PIXMIX can use numerous mixing sets, including real images. Results are using CIFAR-100. **Bold** is best, and <u>underline</u> is second best. We compare Fractals + FVis, the mixing set used as PIXMIX's default mixing set, to other datasets from prior work. The 300K Random Images are real images scraped from online for Outlier Exposure. We discover the distinct utility of Fractals and FVis. By utilizing the 300K Random Images mixing set, PIXMIX can attain a 19.6% error rate, though fractals can provide more robustness than these real images.

Ablations - Mixing Strategies

	Accuracy	Corruptions	Consistency	Adversaries	Calibration	Anomaly
	Clean Error (↓)	C mCE (↓)	CIFAR-P mFR (↓)	PGD Error (↓)	C RMS (↓)	Detection AUROC (↑)
PIXMIX Original	20.3	30.5	5.7	92.9	8.1	89.3
Mix Input	19.9	34.1	6.4	96.7	15.5	86.5
Mix Aug	20.6	31.1	6.2	94.2	6.0	89.7
Iterative	21.1	31.4	5.6	90.6	12.7	86.7

Table 5. PIXMIX variations on CIFAR-100. Mix Input only mixes with augmented versions of the clean image. Mix Aug only mixes with images from the mixing set (i.e. fractals and feature visualizations). Iterative mixes with feature visualizations computed on the fly for the current network. Using the mixing set alone is more effective than augmented images alone, and combining them can further improve performance on several metrics.

- Mix Input : only includes clean images in the mixing pipeline and does not use mixing set at all
- Mix Aug : only mixes with images from the mixing set
- Iterative : mixes with feature visualizations computed on the fly for the network being trained

Ablations - Hyperparameter Sensitivity

	$k = 2$	$k = 3$	$k = 4$
$\beta = 5$	20.2 31.6	20.0 31.1	20.1 30.8
$\beta = 4$	19.7 31.3	20.3 30.9	20.1 30.7
$\beta = 3$	20.3 31.2	20.2 30.7	20.3 30.5

Table 14. Performance is not strongly affected by hyperparameters. We include the CIFAR-100 test set error and the CIFAR-100-C mCE for each hyperparameter setting.

- Performance is very stable across a range of hyperparameters

Intuitions

Learning to See by Looking at Noise

Manel Baradad*
MIT CSAIL
mbaradad@mit.edu

Jonas Wulff*
MIT CSAIL
jwulff@csail.mit.edu

Tongzhou Wang
MIT CSAIL
tongzhou@mit.edu

Phillip Isola
MIT CSAIL
phillipi@mit.edu

Antonio Torralba
MIT CSAIL
torralba@mit.edu

Abstract

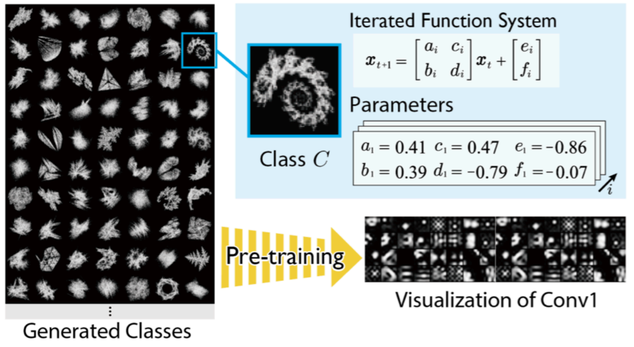
Current vision systems are trained on huge datasets, and these datasets come with costs: curation is expensive, they inherit human biases, and there are concerns over privacy and usage rights. To counter these costs, interest has surged in learning from cheaper data sources, such as unlabeled images. In this paper, we go a step further and ask if we can do away with real image datasets entirely, by learning from procedural noise processes. We investigate a suite of image generation models that produce images from simple random processes. These are then used as training data for a visual representation learner with a contrastive loss. In particular, we study statistical image models, randomly initialized deep generative models, and procedural graphics models. Our findings show that it is important for the noise to capture certain structural properties of real data but that good performance can be achieved even with processes that are far from realistic. We also find that diversity is a key property for learning good representations.

- Specific dataset > FractalDB
- Structural complexity and diversity are key properties for good downstream transfer

Pre-training without Natural Images

Hirokatsu Kataoka · Kazushige Okayasu · Asato Matsumoto · Eisuke Yamagata · Ryosuke Yamada · Nakamasa Inoue · Akio Nakamura · Yutaka Satoh

Abstract Is it possible to use convolutional neural networks pre-trained without any natural images to assist natural image understanding? The paper proposes a novel concept, Formula-driven Supervised Learning. We automatically generate image patterns and their category labels by assigning fractals, which are based on a natural law existing in the background knowledge of the real world. Theoretically, the use of automatically generated images instead of natural images in the pre-training phase allows us to generate an infinite scale dataset of labeled images. Although the models pre-trained with the proposed Fractal DataBase (FractalDB), a database without natural images, does not necessarily outperform models pre-trained with human annotated datasets at all settings, we are able to partially surpass the accuracy of ImageNet/Places pre-trained models. The image representation with the proposed FractalDB captures a unique feature in the visualization of convolutional layers and attentions.[1]



- FractalDB pre-trained model > ImageNet pre-trained model on some downstream task
- FractalDB : algorithmically generated fractal images

Why Learning Fractals Help?

Nonaccidental properties underlie human categorization of complex natural scenes

Dirk B Walther [1], Dandan Shen

Affiliations + expand

PMID: 24474725 PMCID: PMC3984348 DOI: 10.1177/0956797613512662

Free PMC article

Abstract

Humans can categorize complex natural scenes quickly and accurately. Which scene properties enable people to do this with such apparent ease? We extracted structural properties of contours (orientation, length, curvature) and contour junctions (types and angles) from line drawings of natural scenes. All of these properties contain information about scene categories that can be exploited computationally. However, when we compared error patterns from computational scene categorization with those from a six-alternative forced-choice scene-categorization experiment, we found that only junctions and curvature made significant contributions to human behavior. To further test the critical role of these properties, we perturbed junctions in line drawings by randomly shifting contours and found a significant decrease in human categorization accuracy. We conclude that scene categorization by humans relies on curvature as well as the same nonaccidental junction properties used for object recognition. These properties correspond to the visual features represented in area V2.

‘Non-accidental’ properties that humans may use, namely “structural properties of contours (orientation, length, curvature) and contour junctions (types, angles) from line drawings of natural scenes”.

Fractals posses some of these structural properties, and they are highly non-accidental and unlikely to arise from maximum entropy, unstructured random noise processes.

Gaussian noise increase input entropy, such noise has maximal descriptive complexity but introduce little structural complexity.

Why Learning Fractals Help?

Pretrained Transformers As Universal Computation Engines

Kevin Lu
UC Berkeley
kzl@berkeley.edu

Aditya Grover
Facebook AI Research
adityagrover@fb.com

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

Igor Mordatch
Google Brain
imordatch@google.com

Abstract

We investigate the capability of a transformer pretrained on natural language to generalize to other modalities with minimal finetuning – in particular, without finetuning of the self-attention and feedforward layers of the residual blocks. We consider such a model, which we call a Frozen Pretrained Transformer (FPT), and study finetuning it on a variety of sequence classification tasks spanning numerical computation, vision, and protein fold prediction. In contrast to prior works which investigate finetuning on the same modality as the pretraining dataset, we show that pretraining on natural language can improve performance and compute efficiency on non-language downstream tasks. Additionally, we perform an analysis of the architecture, comparing the performance of a random initialized transformer to a random LSTM. Combining the two insights, we find language-pretrained transformers can obtain strong performance on a variety of non-language tasks[1].

Model	Bit Memory	XOR	ListOps	MNIST	C10	C10 LRA	Homology
FPT	100%	100%	38.4%	98.0%	68.2%	38.6%	12.7%
Random	75.8%	100%	34.3%	91.7%	61.7%	36.1%	9.3%
Bit	100%	100%	35.4%	97.8%	62.6%	36.7%	7.8%
ViT	100%	100%	37.4%	97.8%	72.5%	43.0%	7.5%

Table 2: Test accuracy of language-pretrained (FPT) vs randomly initialized (Random) vs Bit Memory pretraining (Bit) vs pretrained Vision Transformer (ViT) models. The transformer is frozen.