ROBUSTNESS OF NEURAL NETWORKS

Presentation · April 2022

DOI: 10.13140/RG.2.2.23891.25123

CITATIONS
0

READS
2

1 author:

Sergey Tarasenko
Tata Consultancy Services Limited
34 PUBLICATIONS **30** CITATIONS

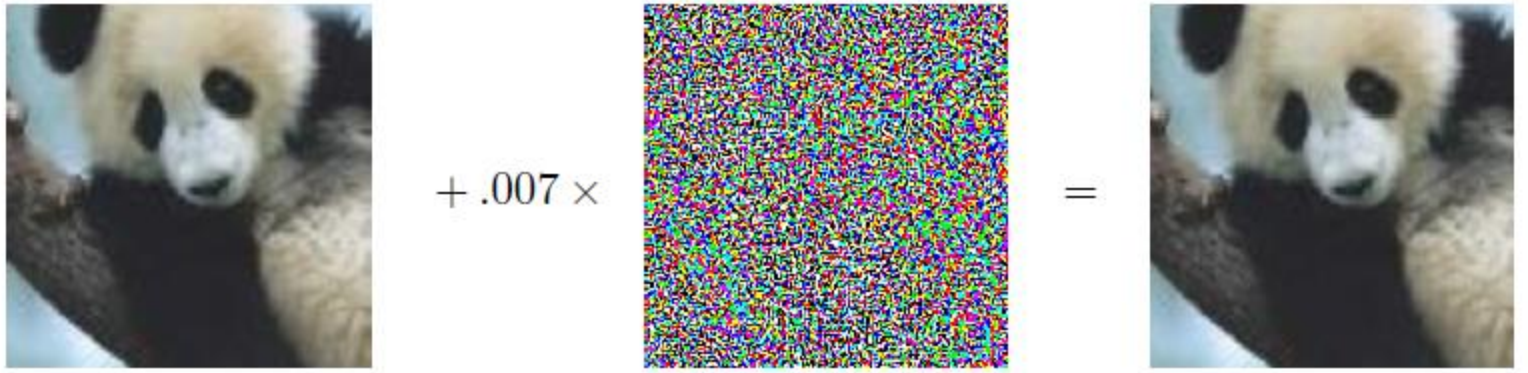
SEE PROFILE

ROBUSTNESS OF NEURAL NETWORKS

SERGEY TARASENKO, PHD

EXAMPLES OF ERRONEOUS BEHAVIOR BY NEURAL NETWORKS

EXAMPLES OF ERRONEOUS BEHAVIOR :: MISCLASSIFICATION



Source: Goodfellow et al. (2015)

EXAMPLES OF ERRONEOUS BEHAVIOR :: INCORRECT REGRESSION

1.1 original 1.2 with added rain

Figure 1: A sample dangerous erroneous behavior found by DeepTest in the *Chauffeur* DNN. *Source:* Tian et al. (2018)

(a) Input 1 (b) Input 2 (darker version of 1)

Figure 1: An example erroneous behavior found by DeepXplore in Nvidia DAVE-2 self-driving car platform. The DNN-based self-driving car correctly decides to turn left for image (a) but incorrectly decides to turn right and crashes into the guardrail for image (b), a slightly darker version of (a). *Source:* Kei et al. (2017)

In these examples, DNS outputs two values:

- Direction (binary value)
- Steering Angle (numeric value)

Arrow corresponds to the value of steering angle

EXAMPLES OF ERRONEOUS BEHAVIOR :: INCORRECT SEGMENTATION

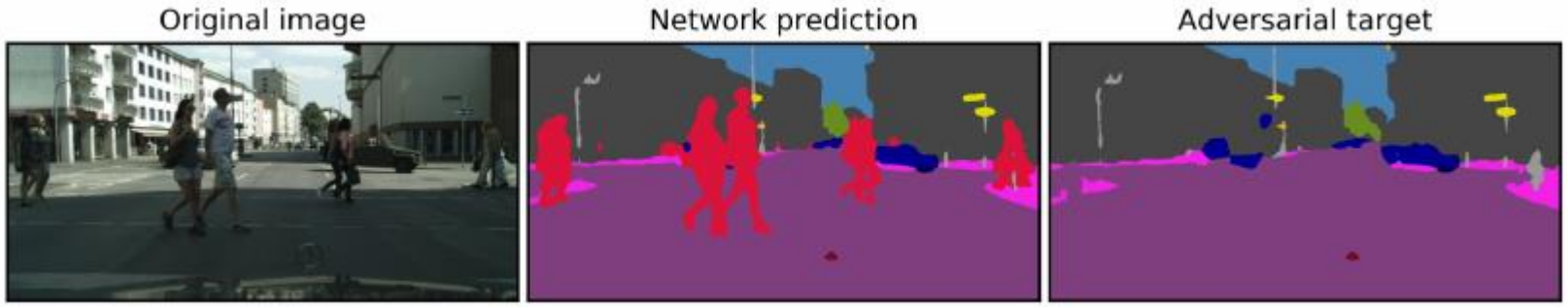


Figure 7: Adversary examples of hiding pedestrians in the semantic segmentation task [112]. Left image: original image; Middle image: the segmentation of the original image predicted by DNN; Right image: the segmentation of the adversarial image predicted by DNN.

Source: Yuan et al. (2019)

EXAMPLES OF ERRONEOUS BEHAVIOR :: INCORRECT OBJECT DETECTION

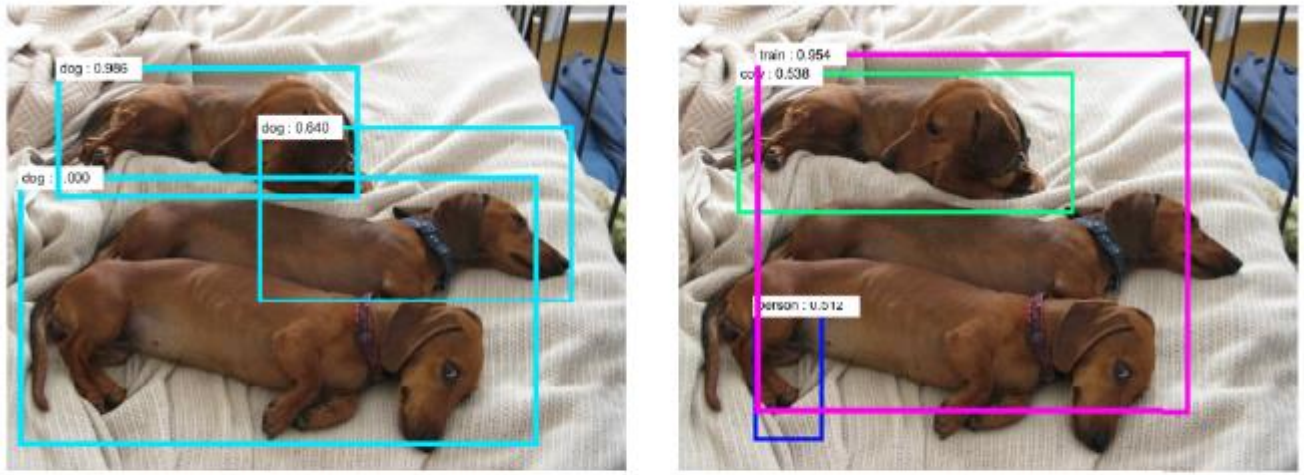


Figure 6: An adversarial example for object detection task [22]. Left: object detection on a clean image. Right: object detection on an adversarial image.

Source: Yuan et al. (2019)

ROBUSTNESS DEFINITION

ROBUSTNESS DEFINITIONS :: OTHER DEFINITIONS

Consider a classifier $f : \mathcal{X} \to \mathcal{Y}$ train on samples from distribution $\mathcal{D}$ and set of corruption function C . Then we define accuracy of the classifier as $\mathbb{P}_{(x,y)\sim\mathcal{D}}(f(x) = y)$

- The corruption robustness is $\mathbb{E}_{c\sim C}[\mathbb{P}_{(x,y)\sim\mathcal{D}}(f(c(x) = y))]$. It measures the classifier's average-case performance on corruption C
- Adversarial Robustness $\min_{\|\delta\|_p<b} \mathbb{P}_{(x,y)\sim\mathcal{D}}(f(x+\delta) = y)$, b is a small budget, δ is a perturbation. It measures the worst case performance on small, additive, classifier tailored perturbations

Source: Hendrycks and Dietterich (2019)

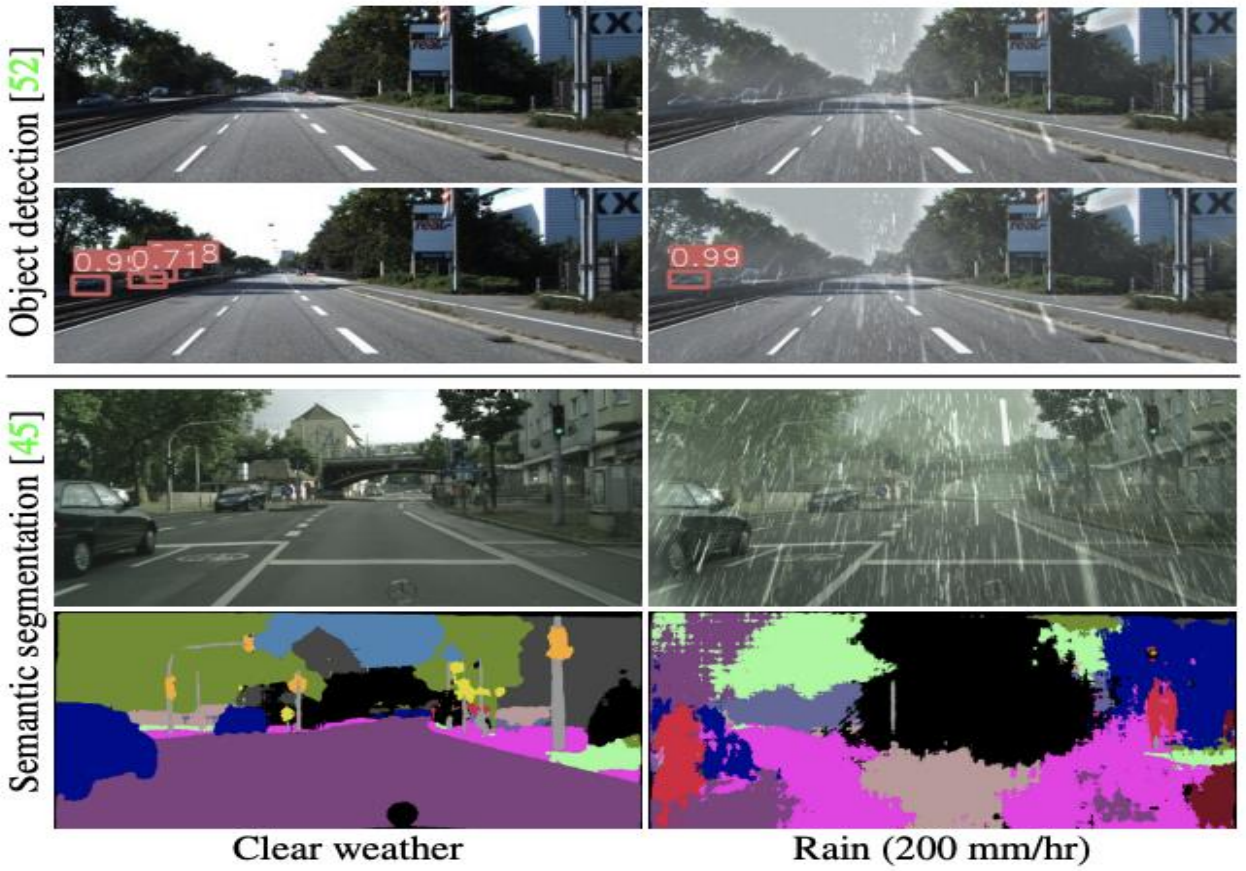
SOURCES OF INSTABILITY

SOURCES OF INSTABILITY

- Weather condition / Atmospheric Events / Day-Night Cycle
- Noise
- Adversarial Attacks
- Different distributions between Training Dataset and Actual Data

SOURCES OF INSTABILITY :: WEATHER ETC.

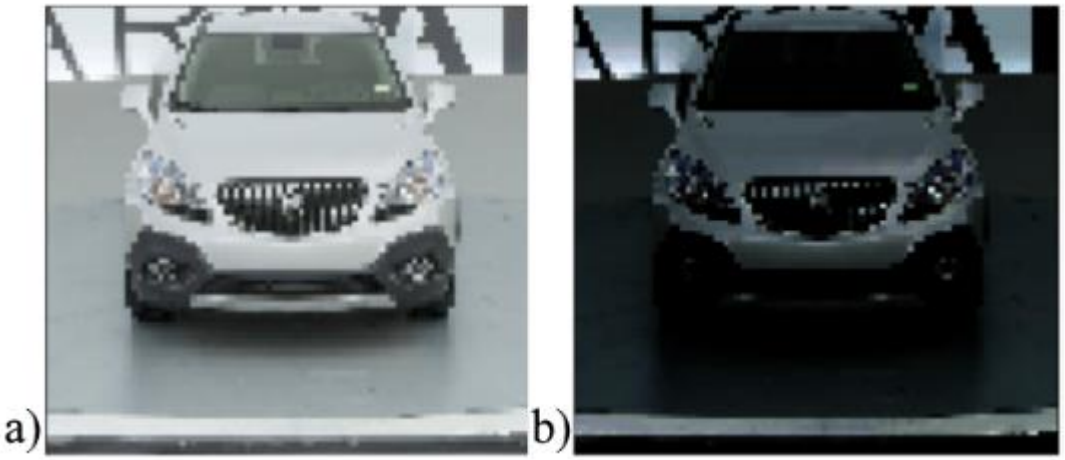
- Weather condition



Source: Halder at el. (2019)

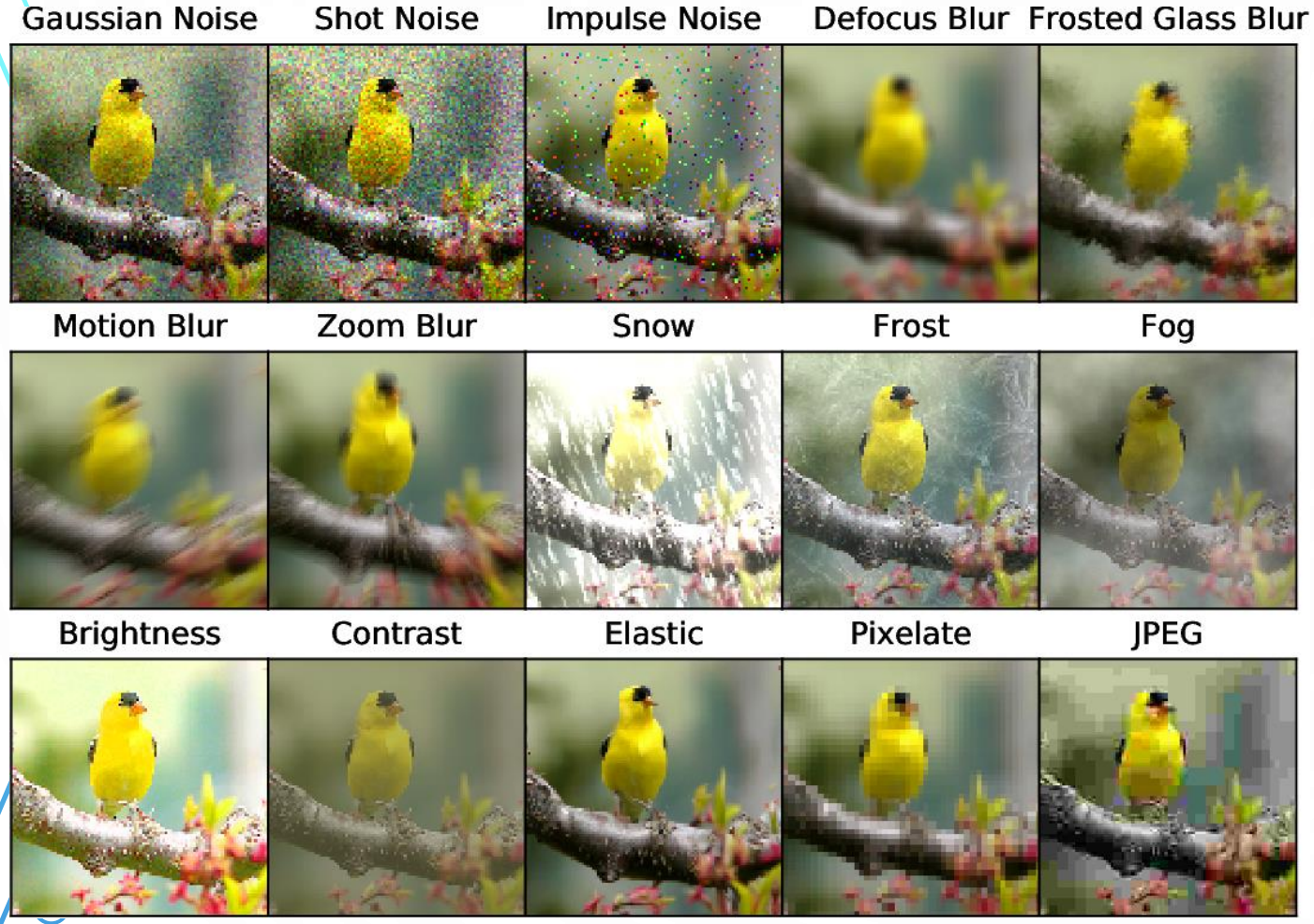
SOURCES OF INSTABILITY :: WEATHER ETC.

- Day-Night Cycle



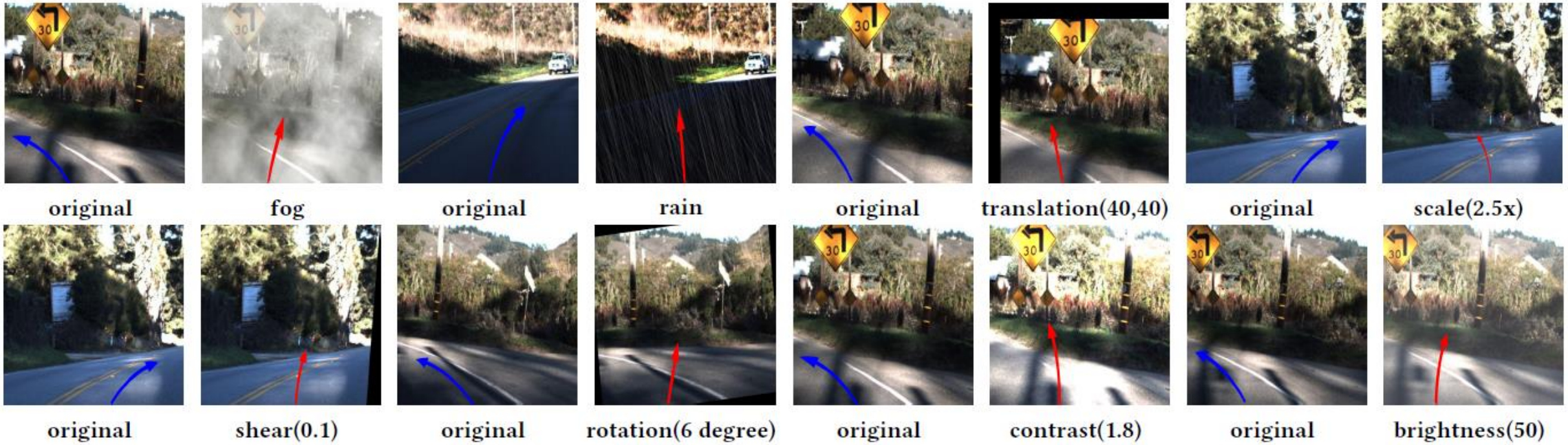
Source: Tarasenko and Takahashi (2020b)

SOURCES OF INSTABILITY :: NOISE



Source: Hendrycks and Dietterich (2019)

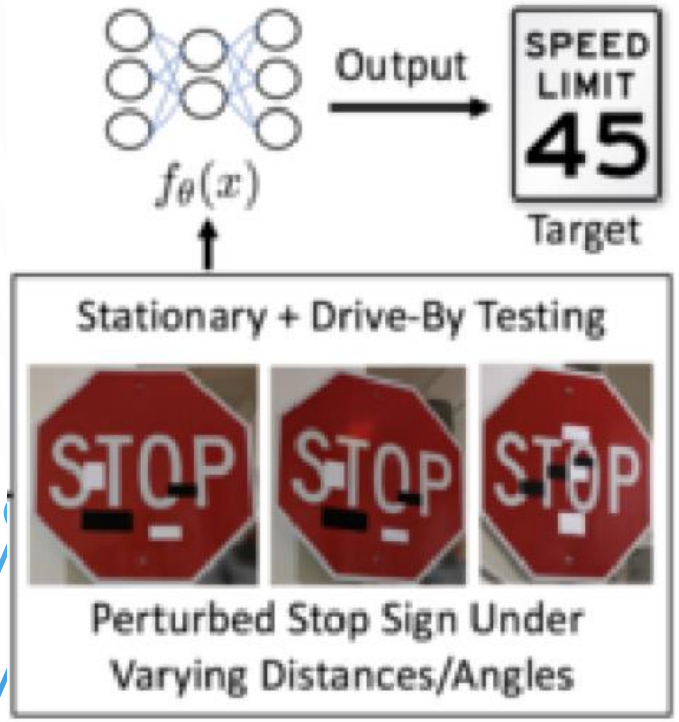
SOURCES OF INSTABILITY :: NOISE



Source: Tian et al. (2018)

SOURCES OF INSTABILITY :: ADVERSARIAL ATTACKS

- An adversarial image is a clean image perturbed by a small distortion carefully crafted to confuse a classifier.



Source: Eykholt et al. (2018)

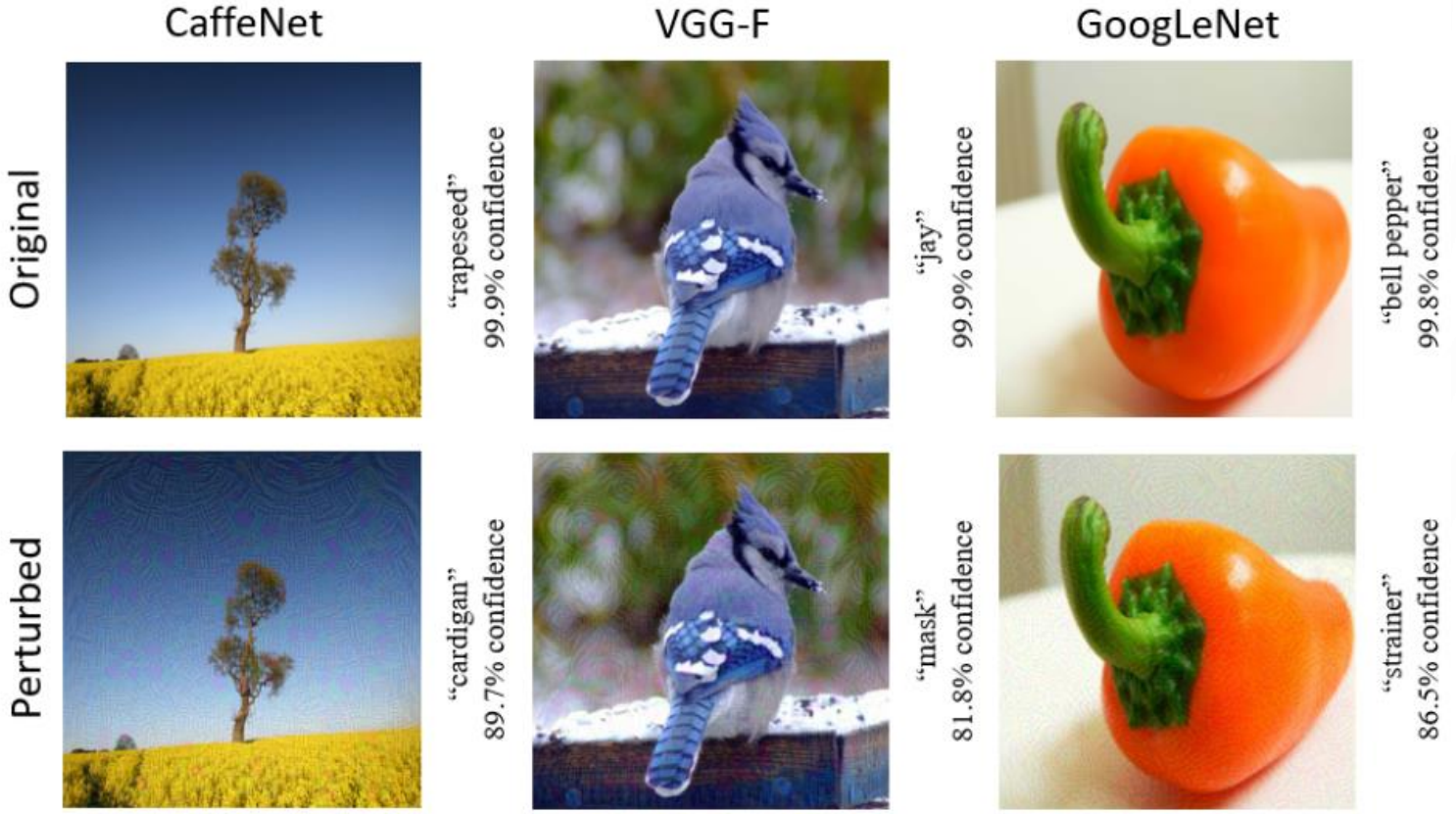


Fig. 1: Example of attacks on deep learning models with 'universal adversarial perturbations' [16]: The attacks are shown for the CaffeNet [9], VGG-F network [17] and GoogLeNet [18]. All the networks recognized the original clean images correctly with high confidence. After small perturbations were added to the images, the networks predicted wrong labels with similar high confidence. Notice that the perturbations are hardly perceptible for human vision system, however their effects on the deep learning models are catastrophic.

Source: Akhtar and Mian (2018)

SOURCES OF INSTABILITY :: ADVERSARIAL ATTACKS

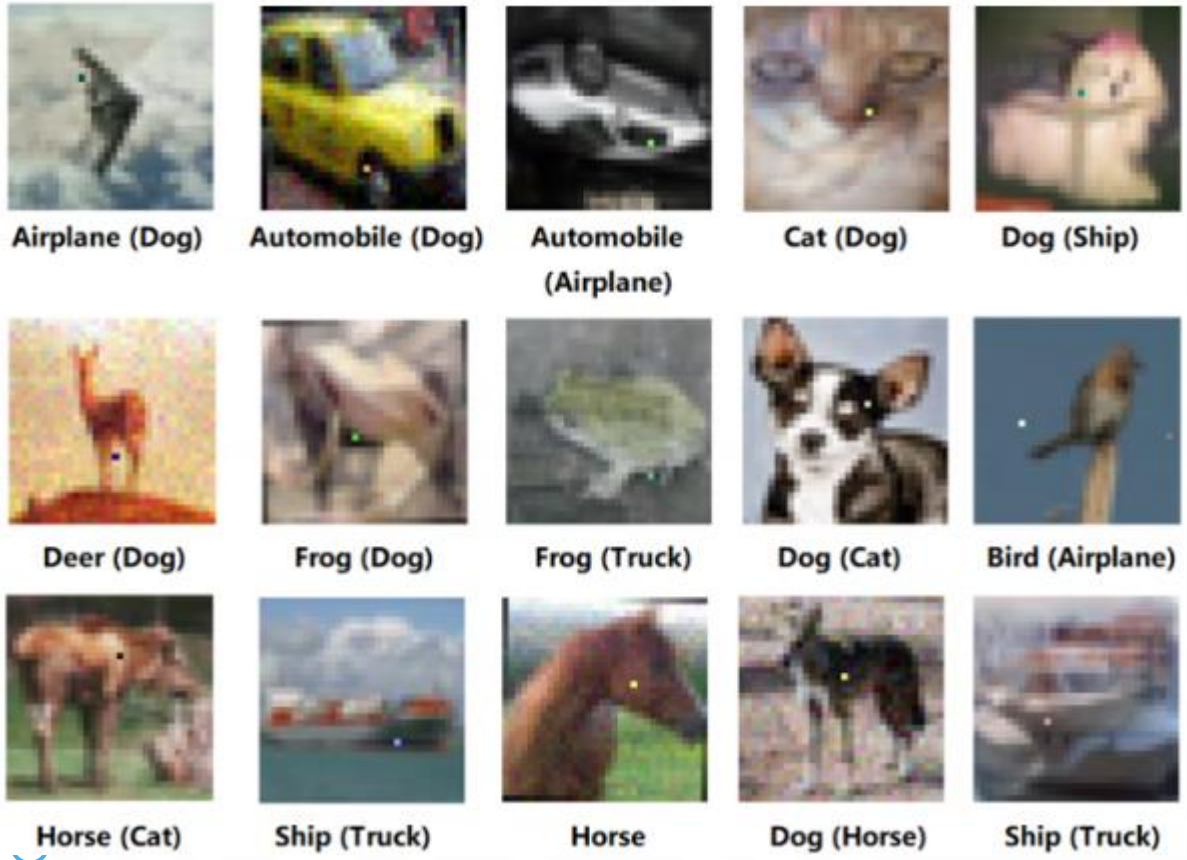


Fig. 3: Illustration of one pixel adversarial attacks [68]: The correct label is mentioned with each image. The corresponding predicted label is given in parentheses.

Source: Akhtar and Mian (2018)

SOURCES OF INSTABILITY :: DATA FROM DIFFERENCE DISTRIBUTIONS

The following issue can impact robustness:

- Scale
- Diversity, representativeness and range of outliers
- Choice of real or synthetic data
- Datasets used specially for robustness testing
- Adversarial and other examples that explore hypothetic domain extremes (corner cases)
- Data drift after model is trained

BENCHMARKING OF NEURAL NETWORKS

BENCHMARKING OF NEURAL NETWORKS :: CLASSIFICATION

A lot of research on creating corrupted datasets has been done by Hendrycks and Dietterich (2019). Here we compare a list of corruptions

Hendrycks and Dietterich (2019)
Noise (Gaussian, Impulse, Shot)
Brightness, Contrast
Blur (motion, defocus, glass, zoom
Weather conditions (snow, frost, fog)
Other (Elastic, Pixelate, JPEG)

BENCHMARKING OF NEURAL NETWORKS :: CORRUPTED DATASETS

- 15 types of Noise, 5 levels of severity for each type of noise

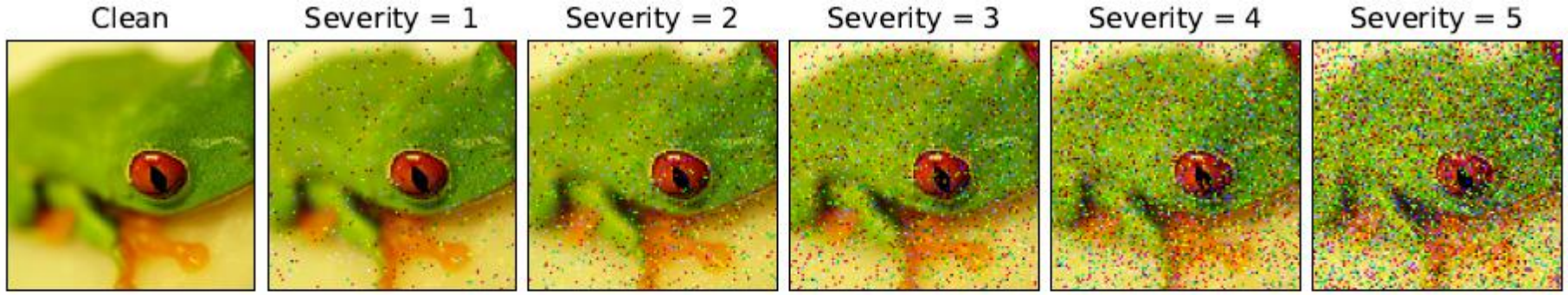


Figure 7: Impulse noise modestly to markedly corrupts a frog, showing our benchmark's varying severities.

Source: Hendrycks and Dietterich (2019)

Figure 6: Pixelation modestly to markedly corrupts a fish, showing our benchmark's varying severities.

Source: Hendrycks et al. (2020)

BENCHMARKING OF NEURAL NETWORKS :: CORRUPTED DATASETS

Using 15 corruptions and 5 levels of severity for each corruption,

Hendrycks and Dietterich (2019), have created 3 new datasets especially to test robustness of the neural networks.

Created Datasets:

- Cifar-10 Corrupted and Cifar-100 Corrupted
- ImageNet-Corrupted

Dataset were created using only Test set of corresponding datasets.

BENCHMARKING OF NEURAL NETWORKS :: METRICS

Robustness metrics to be used on clean and corrupted dataset

- Clean Error is the classification error on the clean or uncorrupted test data.
- In Cifar-10-C and Cifar-100-C datasets, there are 5 levels of severity for each corruption type. It is possible to computed unnormalized Corruption Error (uCE):
- It is possible $\mathrm{uCE}_c = \sum_{s=1}^{5} E_{c,s}$ n error of a given neural network architecture *f* vs. the corruption error of AlexNet

$$\mathrm{CE}_c^f = \left(\sum_{s=1}^{5} E_{s,c}^f\right) \Big/ \left(\sum_{s=1}^{5} E_{s,c}^{\mathrm{AlexNet}}\right)$$

- The average of the 15 corruption errors Mean Corruption Error (mCE)

$$\mathrm{CE}_{\text{Gaussian Noise}}, \mathrm{CE}_{\text{Shot Noise}}, \ldots, \mathrm{CE}_{\text{Pixelate}}, \mathrm{CE}_{\text{JPEG}}$$

- Finally, the amount of additional errors caused by corruptions related to AlexNet performance can be computed as Relative CE and mean Relative CE:

$$\text{Relative CE}_c^f = \left(\sum_{s=1}^{5} E_{s,c}^f - E_{\text{clean}}^f\right) \Big/ \left(\sum_{s=1}^{5} E_{s,c}^{\mathrm{AlexNet}} - E_{\text{clean}}^{\mathrm{AlexNet}}\right)$$

Source: Hendrycks and Dietterich (2019)

BENCHMARKING OF NEURAL NETWORKS :: RESULTS

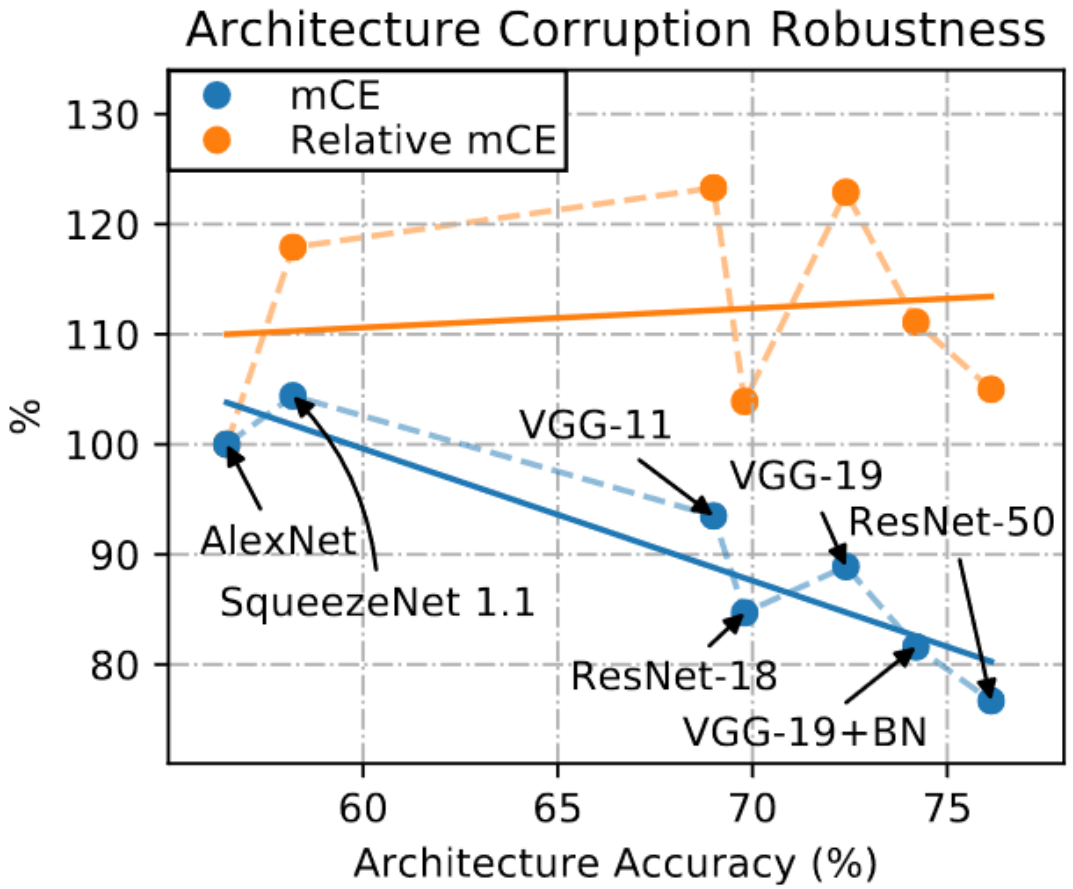


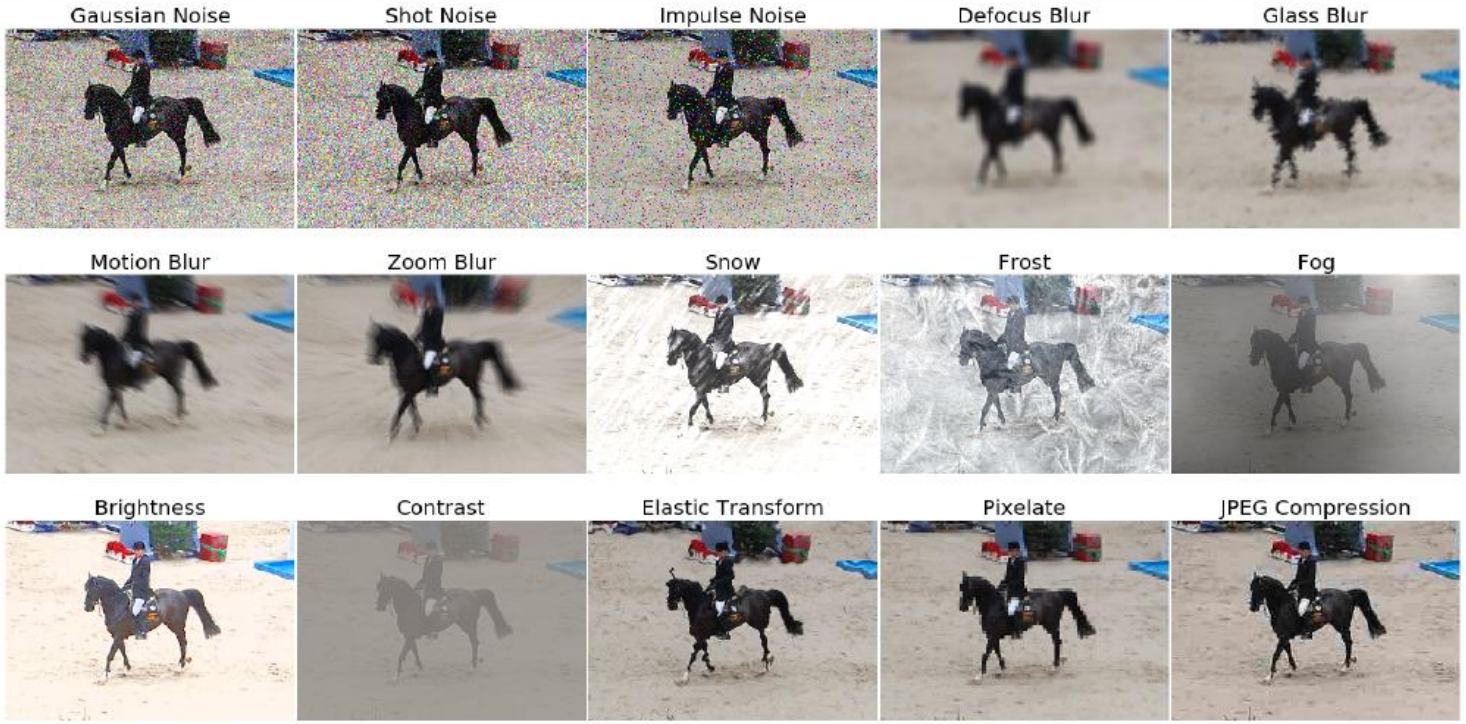
Figure 3: Robustness (mCE) and Relative mCE IMAGENET-C values. Relative mCE values suggest robustness in itself declined from AlexNet to ResNet. "BN" abbreviates Batch Normalization.

Source: Hendrycks and Dietterich (2019)

BENCHMARKING OF NEURAL NETWORKS :: OBJECT DETECTION

Using the methodology proposed by Hendrycks and Dietterich (2019), C. Michaelis el al. (2019) have created 3 corrupted data sets for Object detection:

- Pascal-C
- Coco-C
- CityScapes-C



Source: Michaelis el al. (2019)

BENCHMARKING OF NEURAL NETWORKS :: OBJECT DETECTION

The following metrics were introduced:

- the dataset-specific performance (P) measures

$$\mathrm{P} := \begin{cases} \mathrm{AP}^{50}(\%) & \text{PASCAL VOC} \\ \mathrm{AP}(\%) & \text{MS COCO \& Cityscapes} \end{cases}$$

where AP50 stands for the PASCAL 'Average Precision' metric at 50% Intersection over Union (IoU) and AP stands for the COCO 'Average Precision' metric which averages over IoUs between 50% and 95%.

- On the corrupted data, the benchmark performance is measured in terms of mean performance under corruption (mPC):

$$\mathrm{mPC} = \frac{1}{\mathrm{N}_c} \sum_{c=1}^{\mathrm{N}_c} \frac{1}{\mathrm{N}_s} \sum_{s=1}^{\mathrm{N}_s} \mathrm{P}_{c,s}$$

$P_{c;s}$ is the dataset-specific performance measure evaluated on test data corrupted with corruption *c* under severity level *s* while $N_c = 15$ and $N_s = 5$ indicate the number of corruptions and severity levels, respectively.

- In order to measure relative performance degradation under corruption, the relative performance under corruption (rPC) is

$$\mathrm{rPC} = \frac{\mathrm{mPC}}{\mathrm{P}_{\mathrm{clean}}}$$

Source: Michaelis el al. (2019)

METHODS TO IMPROVE ROBUSTNESS

METHODS TO IMPROVE ROBUSTNESS :: AUGMENTATION TECHNIQUES

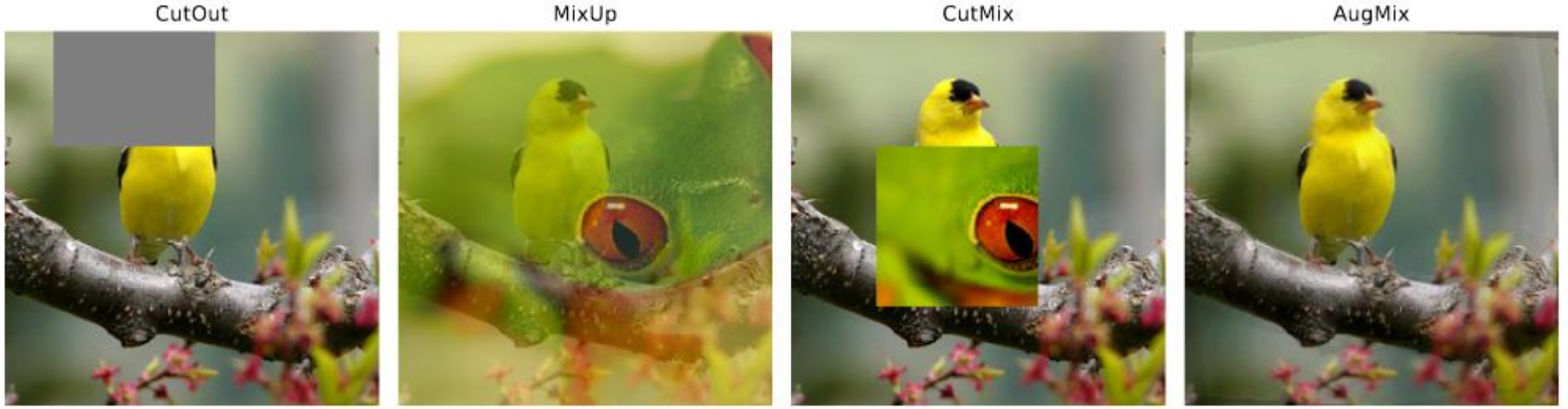
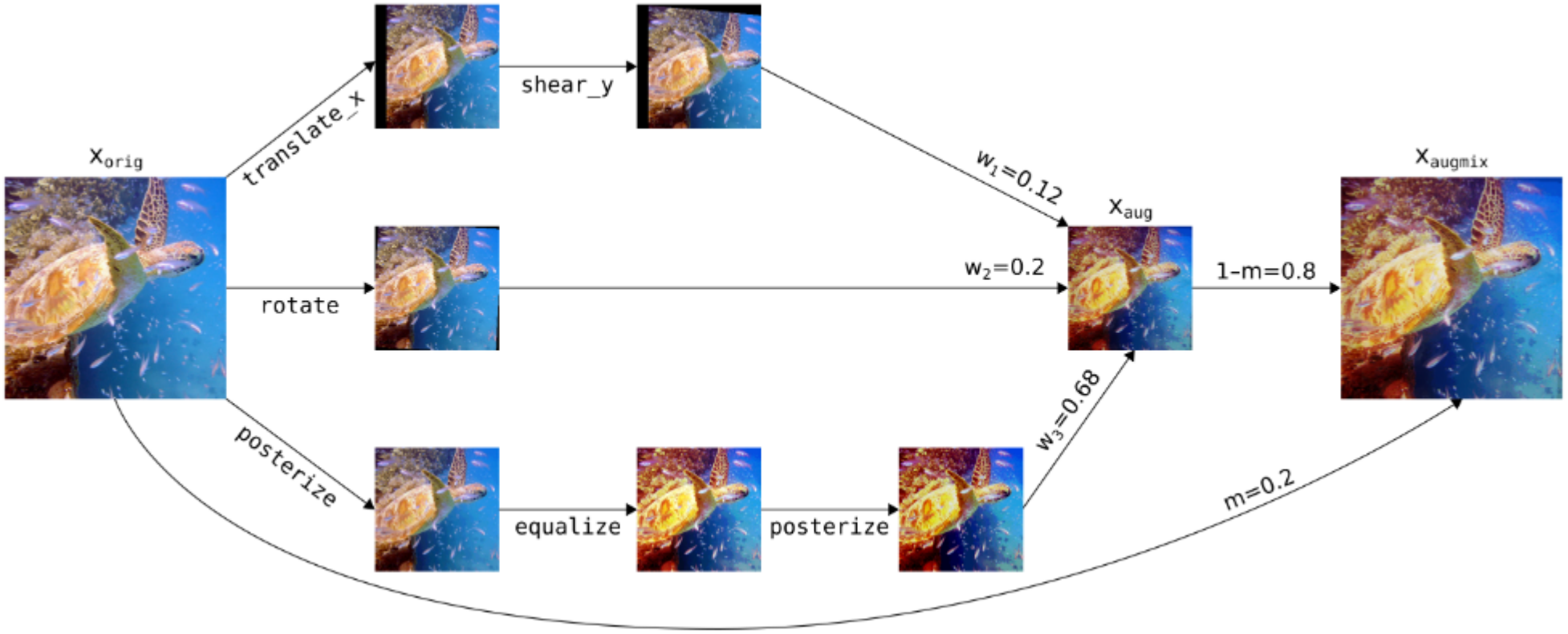


Figure 1: A visual comparison of data augmentation techniques. AUGMIX produces images with variety while preserving much of the image semantics and local statistics.

Source: Hendrycks et al. (2020)

METHODS TO IMPROVE ROBUSTNESS :: AUGMIX



Source: Hendrycks et al. (2020)

METHODS TO IMPROVE ROBUSTNESS :: AUGMIX

Algorithm AUGMIX Pseudocode

Input: Model $\hat{p}$, Classification Loss $\mathcal{L}$, Image x_{orig}, Operations $\mathcal{O} = \{\text{rotate}, \ldots, \text{posterize}\}$
function AugmentAndMix(x_{orig}, $k = 3, \alpha = 1$)
  Fill x_{aug} with zeros
  Sample mixing weights $(w_1, w_2, \ldots, w_k) \sim \text{Dirichlet}(\alpha, \alpha, \ldots, \alpha)$
  for $i = 1, \ldots, k$ **do**
    Sample operations $\text{op}_1, \text{op}_2, \text{op}_3 \sim \mathcal{O}$
    Compose operations with varying depth $\text{op}_{12} = \text{op}_2 \circ \text{op}_1$ and $\text{op}_{123} = \text{op}_3 \circ \text{op}_2 \circ \text{op}_1$
    Sample uniformly from one of these operations chain $\sim \{\text{op}_1, \text{op}_{12}, \text{op}_{123}\}$
    $x_{\text{aug}} \mathrel{+}= w_i \cdot \text{chain}(x_{\text{orig}})$ ▷ *Addition is elementwise*
  end for
  Sample weight $m \sim \text{Beta}(\alpha, \alpha)$
  Interpolate with rule $x_{\text{augmix}} = m x_{\text{orig}} + (1 - m) x_{\text{aug}}$
  return x_{augmix}
end function
$x_{\text{augmix1}} = \text{AugmentAndMix}(x_{\text{orig}})$ ▷ $x_{augmix1}$ *is stochastically generated*
$x_{\text{augmix2}} = \text{AugmentAndMix}(x_{\text{orig}})$ ▷ $x_{augmix1} \neq x_{augmix2}$
Loss Output: $\mathcal{L}(\hat{p}(y \mid x_{\text{orig}}), y) + \lambda \, \text{Jensen-Shannon}(\hat{p}(y \mid x_{\text{orig}}); \hat{p}(y|x_{\text{augmix1}}); \hat{p}(y|x_{\text{augmix2}}))$

Source: Hendrycks et al. (2020)

METHODS TO IMPROVE ROBUSTNESS :: AUGMIX

Test Result of AugMix vs. over augmentation methods using 4 different NN architectures.

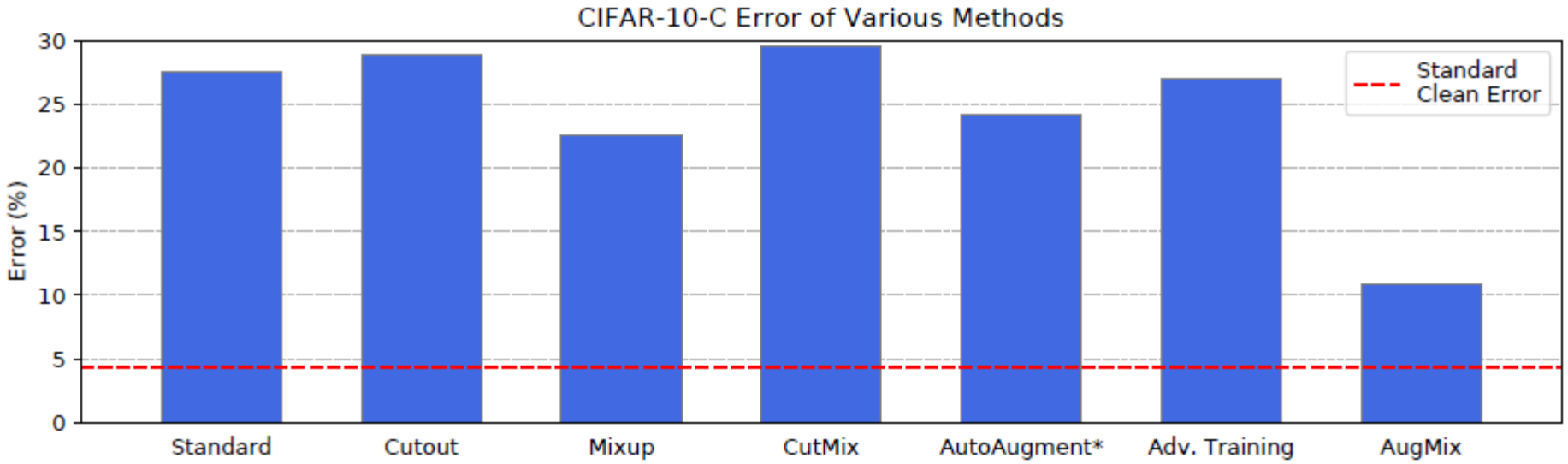


Figure 5: Error rates of various methods on CIFAR-10-C using a ResNeXt backbone. Observe that AUGMIX halves the error rate of prior methods and approaches the clean error rate.

Source: Hendrycks et al. (2020)

METHODS TO IMPROVE ROBUSTNESS :: AUGMIX :: RESULTS

Test Result of AugMix vs. over augmentation methods using 4 different NN architectures. Table below introduces mCE.

		Standard	Cutout	Mixup	CutMix	AutoAugment*	Adv Training	AUGMIX
CIFAR-10-C	AllConvNet	30.8	32.9	24.6	31.3	29.2	28.1	**15.0**
	DenseNet	30.7	32.1	24.6	33.5	26.6	27.6	**12.7**
	WideResNet	26.9	26.8	22.3	27.1	23.9	26.2	**11.2**
	ResNeXt	27.5	28.9	22.6	29.5	24.2	27.0	**10.9**
Mean		29.0	30.2	23.5	30.3	26.0	27.2	**12.5**
CIFAR-100-C	AllConvNet	56.4	56.8	53.4	56.0	55.1	56.0	**42.7**
	DenseNet	59.3	59.6	55.4	59.2	53.9	55.2	**39.6**
	WideResNet	53.3	53.5	50.4	52.9	49.6	55.1	**35.9**
	ResNeXt	53.4	54.6	51.4	54.1	51.3	54.4	**34.9**
Mean		55.6	56.1	52.6	55.5	52.5	55.2	**38.3**

Source: Hendrycks et al. (2020)

METHODS TO IMPROVE ROBUSTNESS :: AUGMIX :: RESULTS

Test Result of AugMix vs. over augmentation methods using 4 different NN architectures. Table below introduces mCE.

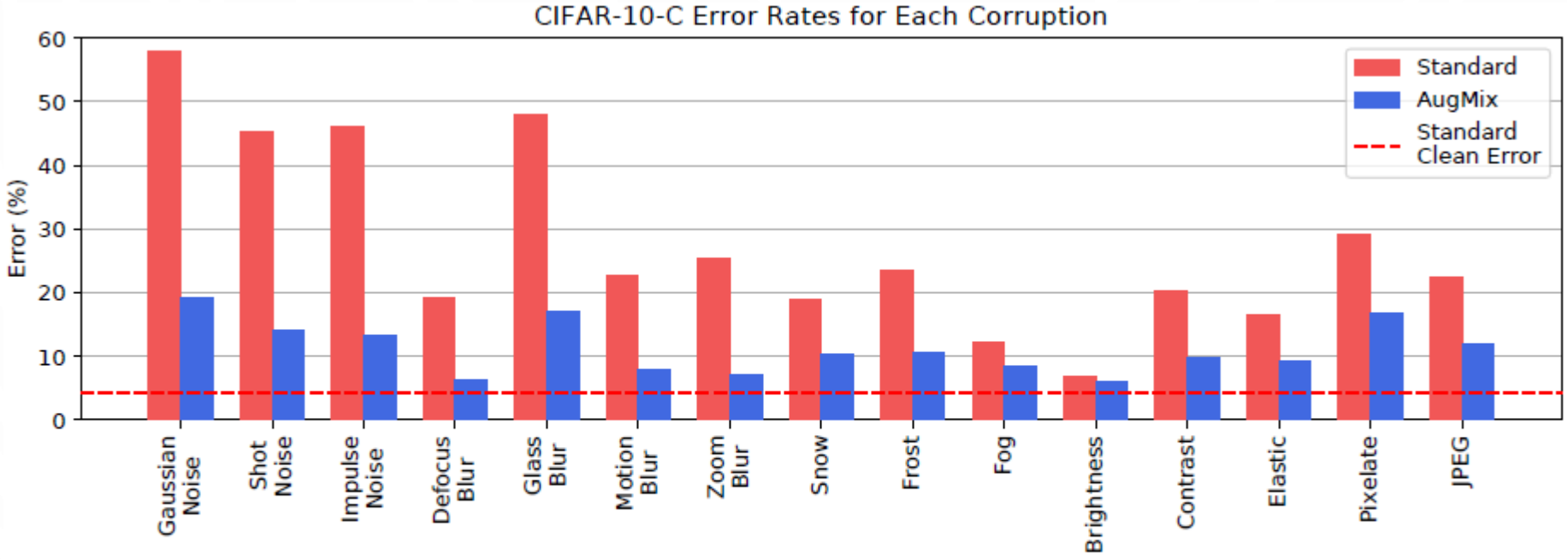


Figure 12: AUGMIX improves corruption robustness across all CIFAR-10-C noise, blur, weather, and digital corruptions, despite the model never having seen these corruptions during training.

Source: Hendrycks et al. (2020)

METHODS TO IMPROVE ROBUSTNESS :: STREAMING NETS (STNETS)

- Most recently Streaming Networks (STNets) have been introduced as a mechanism of robust noise-corrupted images classification. STNets is a family of convolutional neural networks, which consists of multiple neural networks (streams). All streams have different inputs and their outputs are concatenated and fed into a single joint classifier
- The original papers* have illustrated how STNets can successfully classify images from Cifar10, EuroSat and UCmerced datasets, when images were corrupted with various levels of random zero noise

- *
  - https://arxiv.org/abs/1910.11107
  - https://arxiv.org/abs/2004.03334

METHODS TO IMPROVE ROBUSTNESS :: STREAMING NETS (STNETS)

- STNets are constructed of multiple parallel streams. During training, parameters of all streams are tuned independently. Each stream is taking a certain piece of information and its input is different from the inputs of the other streams

- In the original study [https://arxiv.org/abs/2004.03334], each stream had input in the form of the image intensity slice

Complete Image

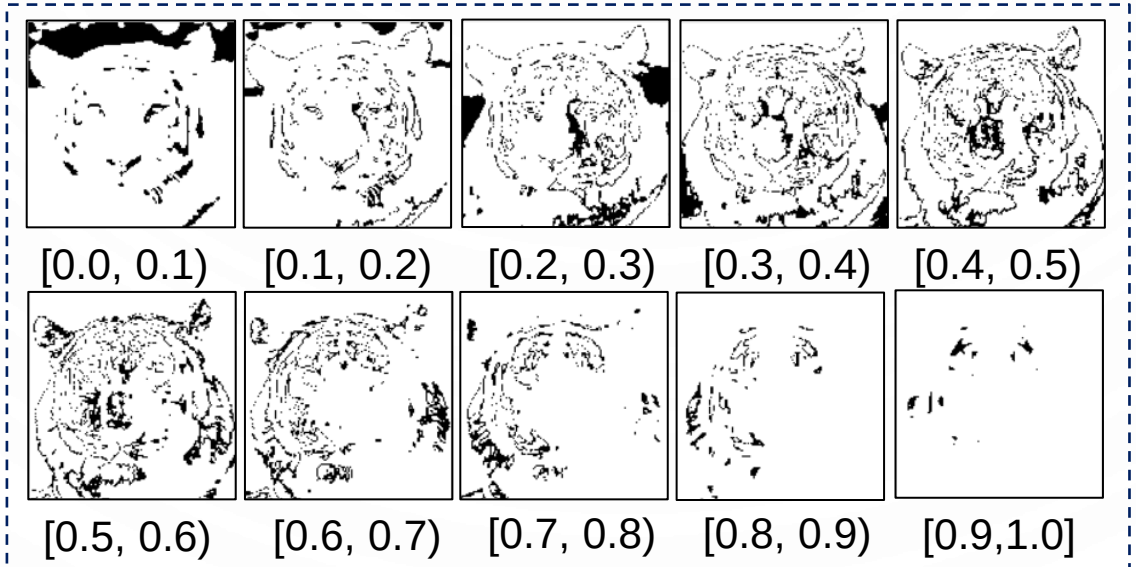
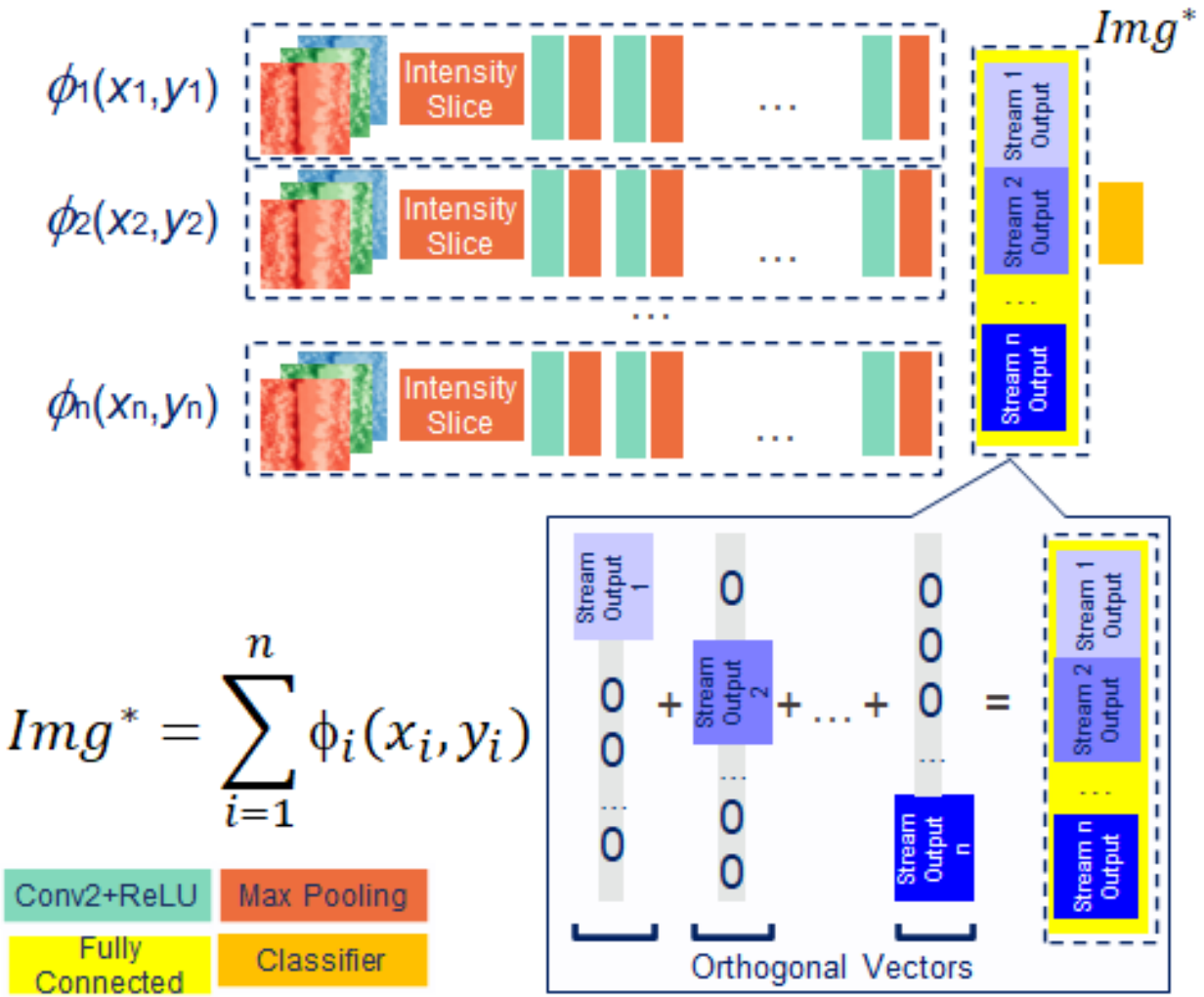


Image Intensity Slices

Source: Tarasenko and Takahashi (2020a)

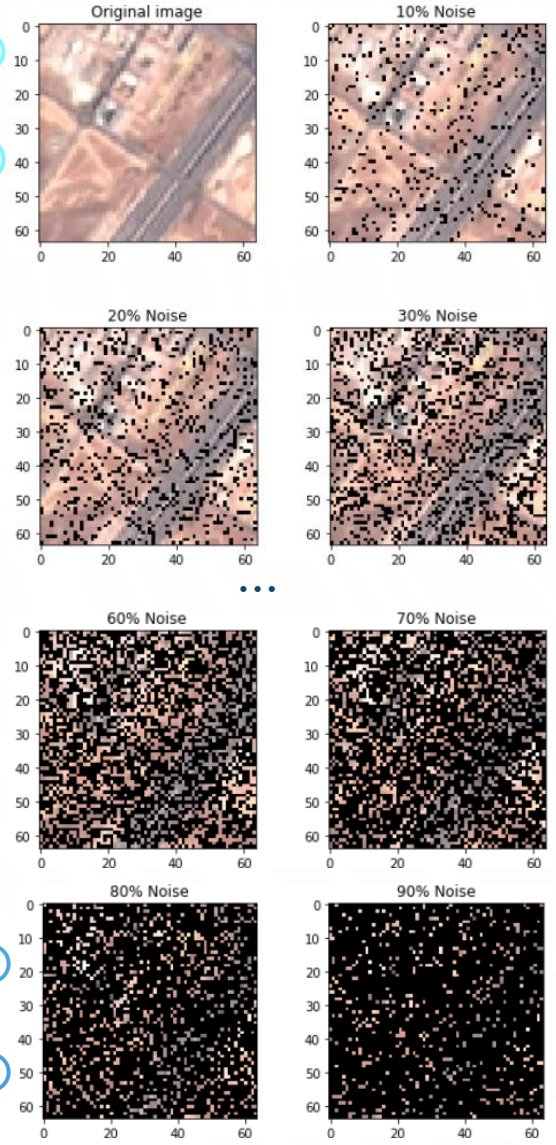
METHODS TO IMPROVE ROBUSTNESS :: STREAMING NETS (STNETS) :: ORTHOGONAL BASIS OF STREAM OUTPUTS

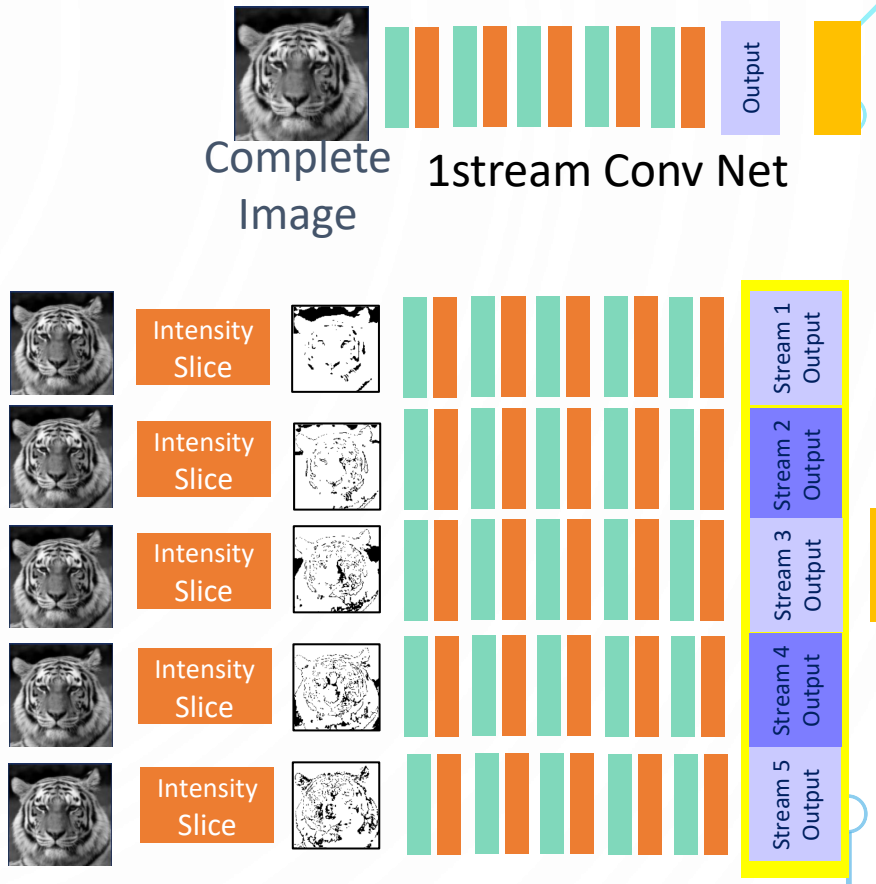


- In STNets stream outputs are concatenated and fed into a classifier
- It is possible to show that input into a classifier is an Orthogonal Basis of the outputs of the STNet's streams

Source: Tarasenko and Takahashi (2020b)

METHODS TO IMPROVE ROBUSTNESS :: STREAMING NETS (STNETS) ::RESULTS





Source: Tarasenko and Takahashi (2020a)

METHODS TO IMPROVE ROBUSTNESS :: STREAMING NETS (STNETS) :: RESULTS :: INCREASE DIVERSITY OF FILTER WEIGHTS

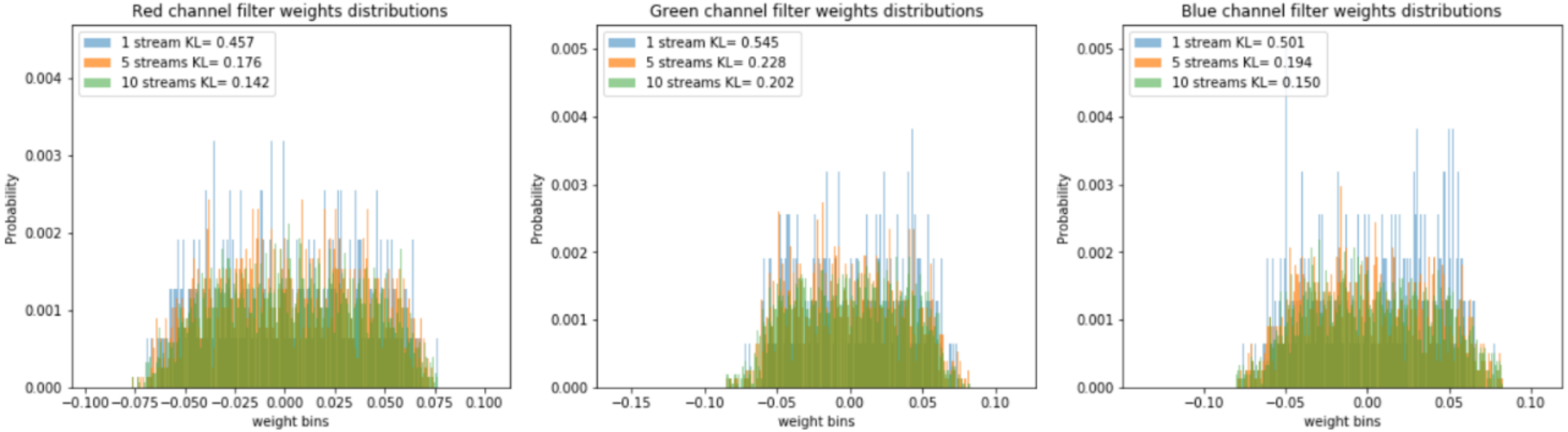
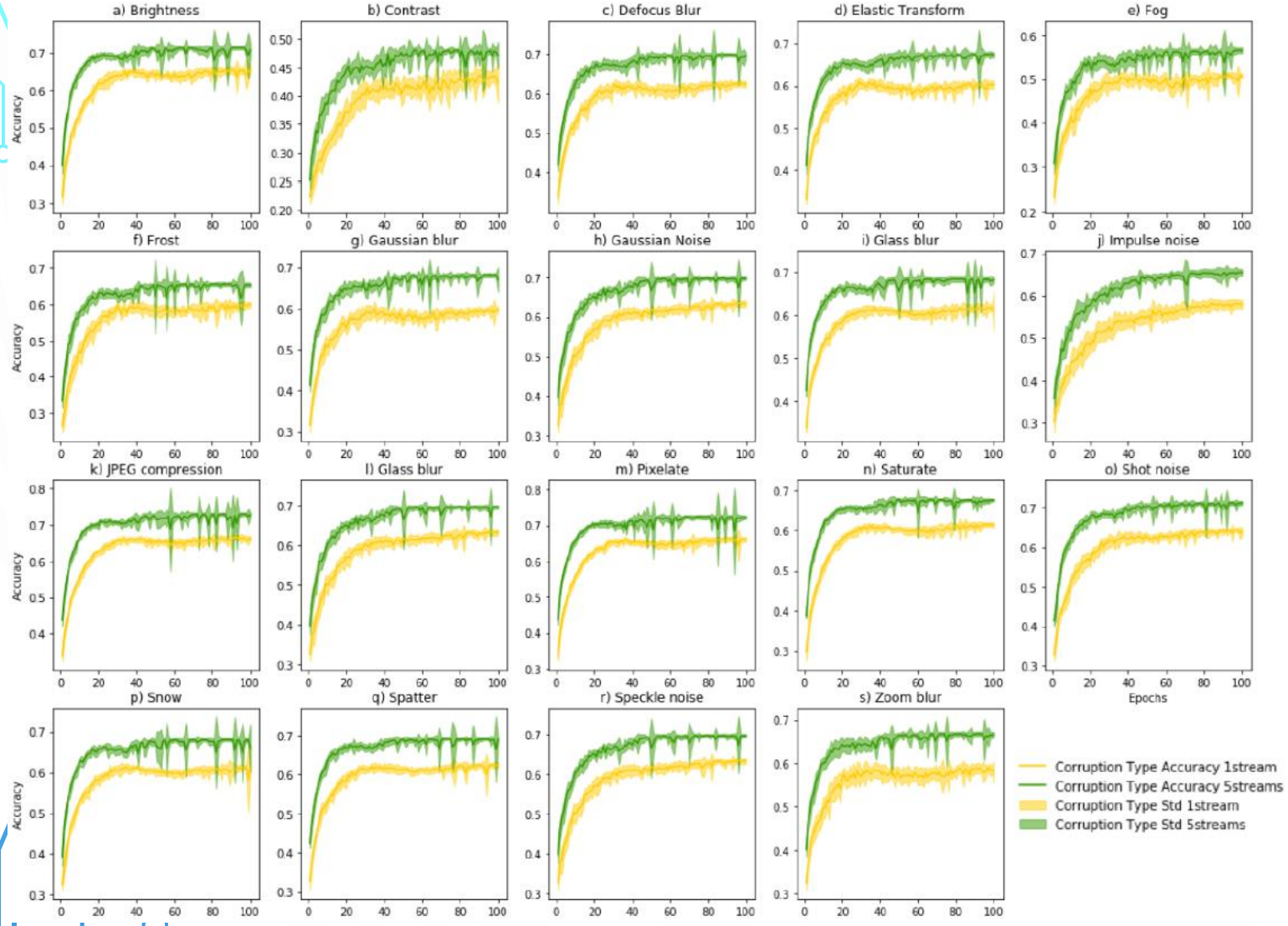
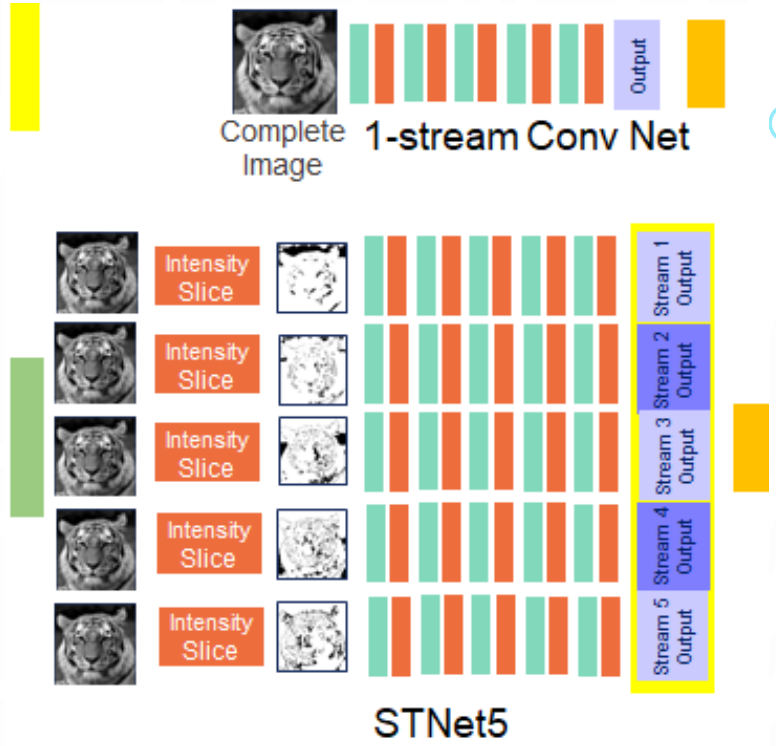


Fig. 28. Test 1: the overall distribution of the filter weights across all the layers and filters of the first conv layers for Eurosat dataset. Notation: "1 stream" implies 1-stream simple conv net, "5 streams" implies 5-stream Streaming Network, "10 streams" implies 10-stream Streaming Network. KL is a short for KullbackLeibler divergence.

Source: Tarasenko and Takahashi (2020a)

STREAMING NETWORKS :: CIFAR-10 CORRUPTED PERFORMANCE

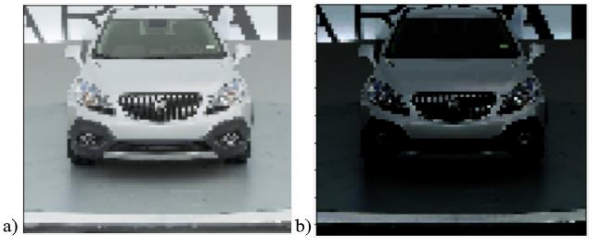




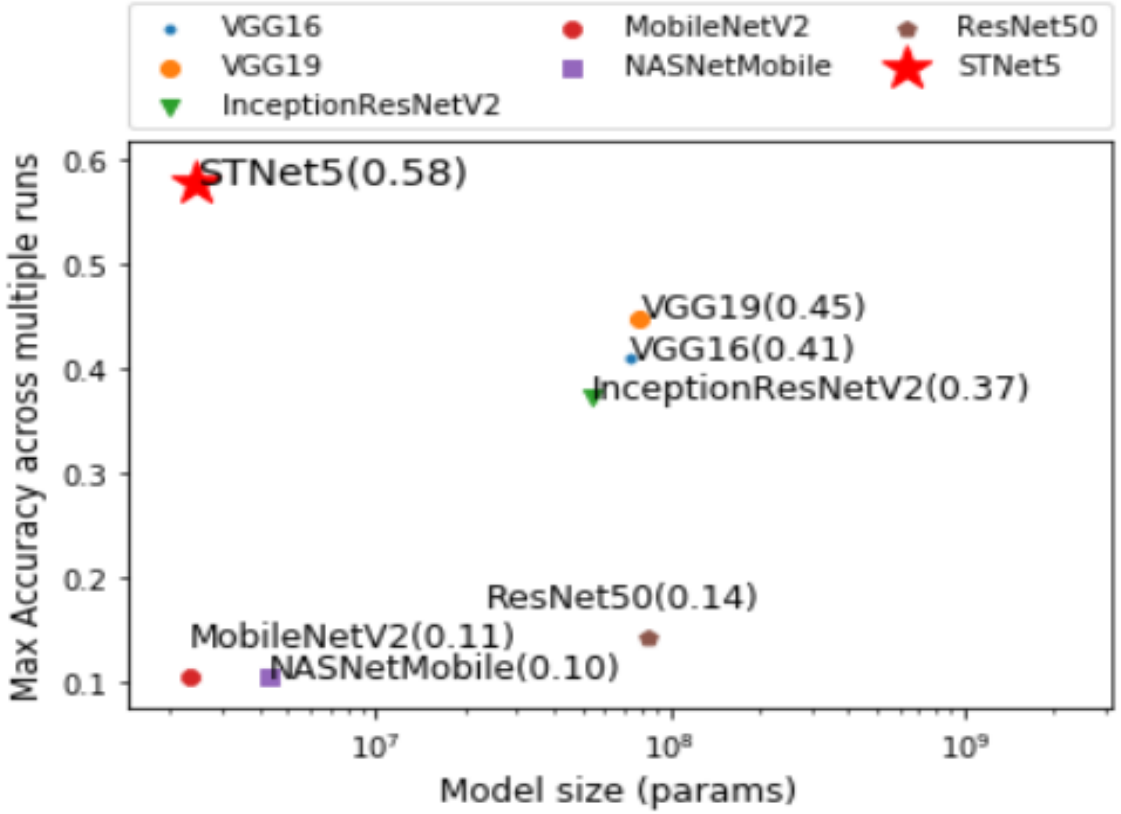
- The results indicate that STNets outperform a baseline 1-stream simple Conv Net for all types of image distortions in terms of test accuracy

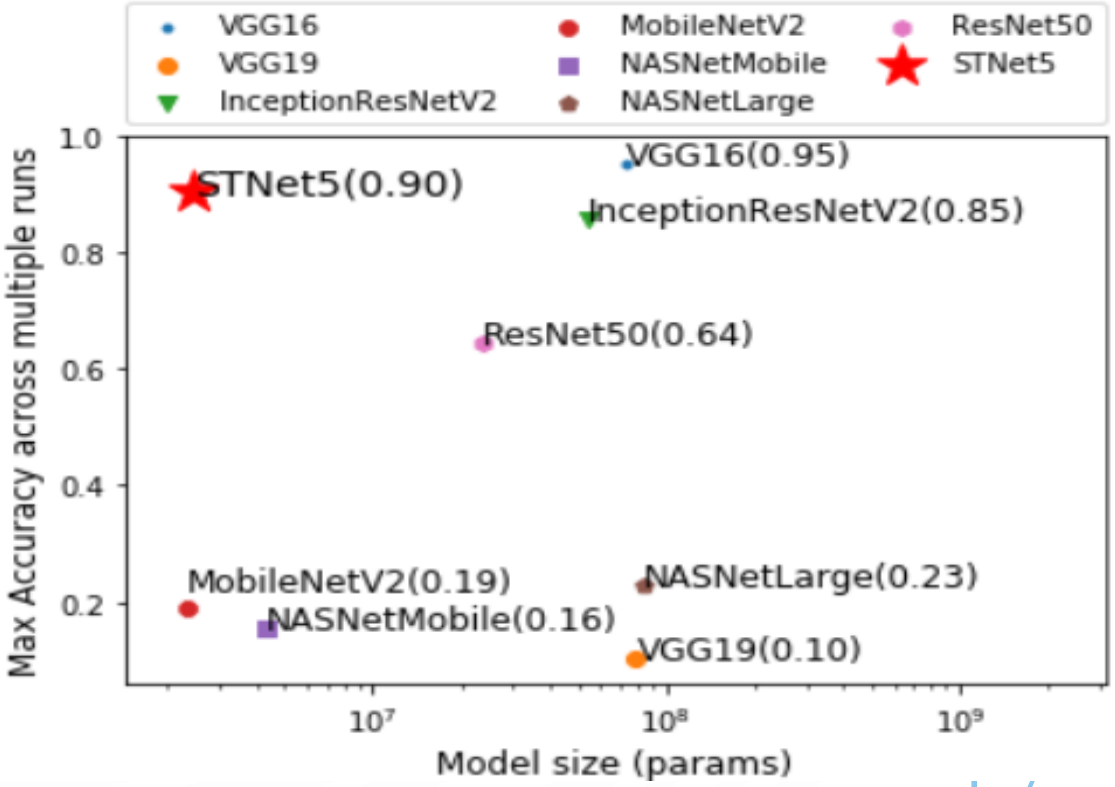
Source: Tarasenko and Takahashi (2020b)

STREAMING NETWORKS :: LOW-LIGHT IMAGES PERFORMANCE



- No data aug: STNet5 outperforms all state-of-the-art model
- Data Aug: STNet5 ranks second. VGG16 is 30x bigger in terms of params.





Source: Tarasenko and Takahashi (2020b)

METHODS TO IMPROVE ROBUSTNESS :: OBJECT DETECTION

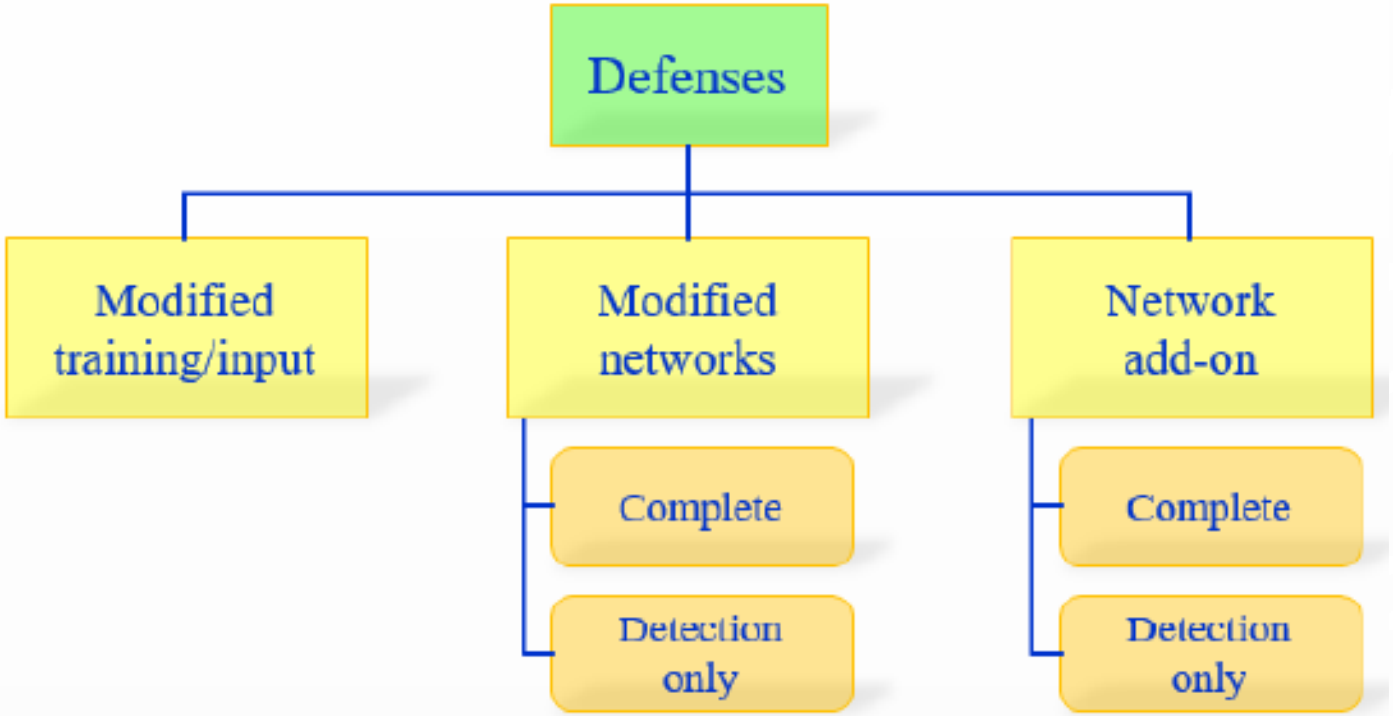
Michaelis el al. (2019) have shown that the following methods increase robustness of Object detection:

- Robustness increase with network capacity
- Training of stylized data increases robustness
- Robustness to natural distortion is connected to synthetic corruption robustness, hence synthetic distortion data can be used to training

Figure 4: Training data visualization for COCO and Stylized-COCO. The three different training settings are: standard data (top row), stylized data (bottom row) and the concatenation of both (termed 'combined' in plots).

Source: Michaelis el al. (2019)

METHODS TO IMPROVE ROBUSTNESS :: DEFENSE FROM ADVERSARIAL ATTACK



Source: Akhtar and Mian (2018)

NEURAL NETWORK TESTING

DEEPTEST AND DEEPXPLORE :: NN AUTOMATED TESTING APPROACH

Despite the tremendous progress, just like traditional software, DNN-based software, including the ones used for autonomous driving, often demonstrate incorrect/unexpected corner-case behaviors that can lead to dangerous consequences like a fatal collision.

Such crashes often happen under rare previously unseen corner-cases. For example, the fatal Tesla crash resulted from a failure to detect a white truck against the bright sky.

The existing mechanisms for detecting such erroneous behaviors depend heavily on manual collection of labeled test data or ad hoc, unguided simulation [11, 20] and therefore miss numerous corner cases.

Since these cars adapt behavior based on their environment as measured by different sensors (e.g., camera, Infrared obstacle detector, etc.), the space of possible inputs is extremely large.

Thus, unguided simulations are highly unlikely to find many erroneous behaviors.

At a conceptual level, these erroneous corner-case behaviors in DNN-based software are analogous to logic bugs in traditional software. Similar to the bug detection and patching cycle in traditional software development, the erroneous behaviors of DNNs, once detected, can be fixed by adding the error-inducing inputs to the training data set and also by possibly changing the model structure/parameters.

functions for similar purposes. These differences make automated testing of DNN-based software challenging by presenting several interesting and novel research problems.

Source: Tian et al. (2018)

DEEPTEST AND DEEPXPLORE :: NN AUTOMATED TESTING APPROACH

Our experience with traditional software has shown that it is hard to build robust safety-critical systems only using manual test cases. Moreover, the internals of traditional software and new DNN-based software are fundamentally different.

For example, unlike traditional software where the program logic is manually written by the software developers, DNN-based software automatically learns its logic from a large amount of data with minimal human guidance. In addition, the logic of a traditional program is expressed in terms of control flow statements while DNNs use weights for edges between different neurons and nonlinear activation functions for similar purposes.

These differences make automated testing of DNN-based software challenging by presenting several interesting and novel research problems.

Source: Tian et al. (2018)

DEEPTEST AND DEEPXPLORE :: NN AUTOMATED TESTING APPROACH

Percularity

- DeepTest and DeepXplore consider that similar to traditional code coverage with tests, neuron coverage in neural network can help finding “bug” or corner-cases (rarely seen examples) for Deep Neural Networks
- Neural coverage is computed for each input. Neuron is considered active if its response exceeds a certain threshold
- Usefulness of the neuron coverage is based on assumption, that all inputs that have similar neuron coverage are members of the same class
- Similar to the code-coverage guided testing tools for traditional software, DeepTest / DeepXplor try to generate inputs that maximize neuron coverage of the test DNN.

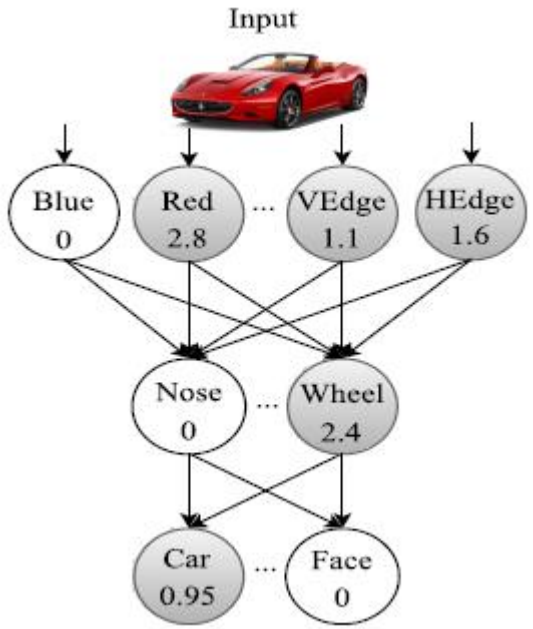
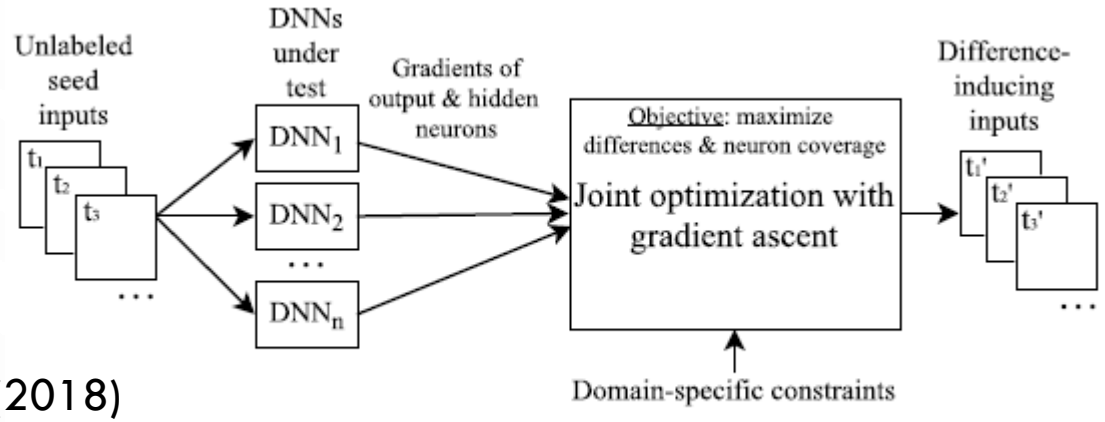


Illustration of neuron coverage: white neurons for **Face**, grey neurons for **Car**

Source: Tian et al. (2018)



Source: Tian et al. (2018)

Figure 5: DeepXplore workflow.

DEEPTEST AND DEEPXPLORE :: NN AUTOMATED TESTING APPROACH

Key features of DeepTest

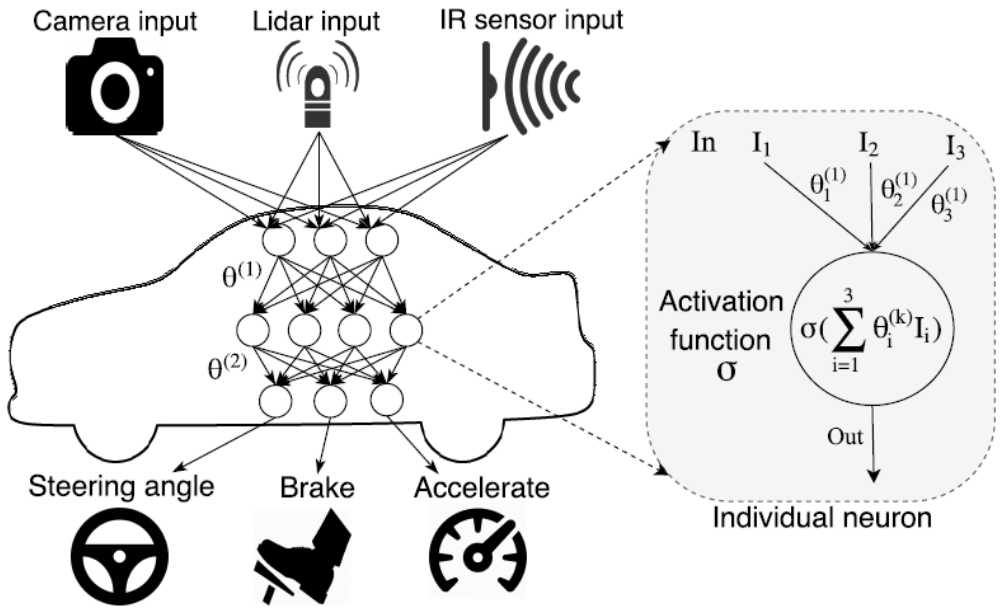
- A systematic technique to automatically synthesize test cases that maximizes neuron coverage in safety-critical DNN-based systems like autonomous cars.
- Empirically demonstrate that changes in neuron coverage correlate with changes in an autonomous car's behavior
- Demonstrate that different realistic image transformations like changes in contrast, presence of fog, etc. can be used to generate synthetic tests that increase neuron coverage.
- Experiments show that the synthetic images can be used for retraining and making DNNs more robust to different corner cases.
- DeepTest found thousands of erroneous behaviors in these systems many of which can lead to potentially fatal collisions.

1.1 original 1.2 with added rain

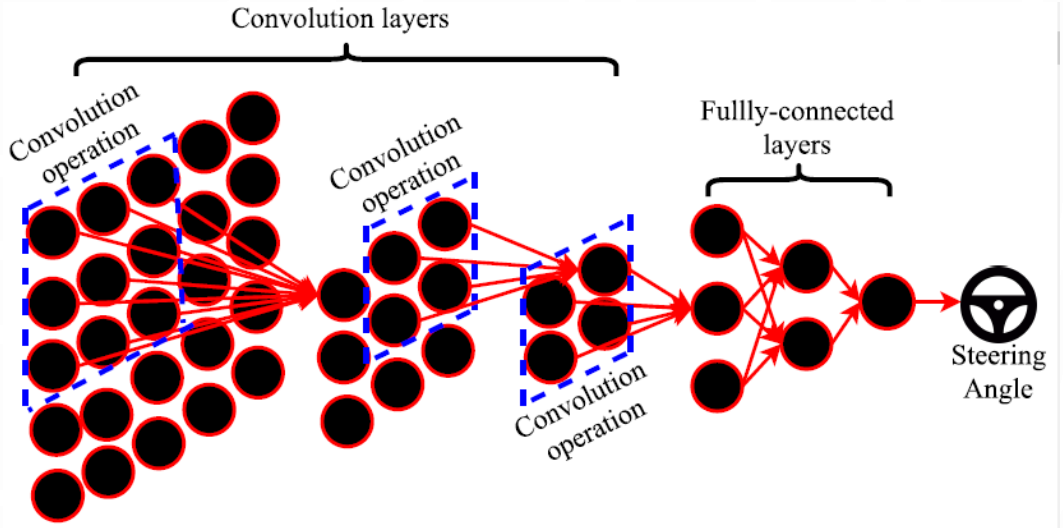
Figure 1: A sample dangerous erroneous behavior found by DeepTest in the *Chauffeur* DNN.

Source: Tian et al. (2018)

DEEPTEST AND DEEPXPLORE :: NN AUTOMATED TESTING APPROACH



Source: Tian et al. (2018)



Source: Tian et al. (2018)

REFERENCES

REFERENCES

1. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow and R. Fergus (2014) Intriguing properties of neural networks, *ICLR 2014*.
2. I. J. Goodfellow, J. Shlens and C. Szegedy (2015) Explaining and Harnessing Adversarial Examples, *ICLR 2015*.
3. K. Pei , Yi. Cao, J. Yang and S. Sekhar Jana (2017) DeepXplore: Automated Whitebox Testing of Deep Learning Systems, Proceedings of the 26th Symposium on Operating Systems Principles.
4. Yu. Tian, K. Pei, S. S. Jana and B. Ray (2018) DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars, IEEE/ACM 40th International Conference on Software Engineering (ICSE) 2018, p. 303-314
5. K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno and D. X. Song (2018) Robust Physical-World Attacks on Deep Learning Visual Classification, CVPR 2018, p. 1625-1634
6. N. Akhtar and A. S. Mian (2018) Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey, IEEE Access, vol. 6, p. 14410-14430.
7. D. Hendrycks and T. G. Dietterich (2019) Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, ICLR 2019.
8. S. S. Halder, J.-F. Lalonde and R. de Charette (2019) Physics-Based Rendering for Improving Robustness to Rain, *ICCV 2019*, p. 10202-10211
9. C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge and W. Brendel (2019) Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming, ArXiv abs/1907.07484.
10. Xi. Yuan, P. He, Q. Zhu and Xi. Li (2019) Adversarial Examples: Attacks and Defenses for Deep Learning, IEEE Transactions on Neural Networks and Learning Systems, vol. 30, p. 2805-2824.
11. D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer and B. Lakshminarayanan (2020) AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty, The International Conference on Learning Representations (ICLR) 2020.
12. S. Tarasenko and F. Takahashi (2020) Streaming Networks: Increase Noise Robustness and Filter Diversity via Hard-wired and Input-induced Sparsity, Arxiv abs/2004.03334
13. S. Tarasenko and F. Takahashi (2020) Streaming Network Applications for Adverse Weather and Lighting Conditions, Vision For All Seasons Workshop, CVPR 2020 [https://vision4allseasons.files.wordpress.com/2020/06/abs_3.pdf]