Seven Myths in Machine Learning Research

Oscar Chang, Hod Lipson

February 2019

Abstract

We present seven myths commonly believed to be true in machine learning research, circa Feb 2019. This is an archival copy of the blog post at https://crazyoscarchang.github.io/2019/02/16/seven-myths-in-machine-learning-research/

Myth 1: TensorFlow is a Tensor manipulation library

It is actually a *Matrix* manipulation library, and this difference is significant.

In Laue et al. [2018], the authors demonstrate that their automatic differentiation library based on actual Tensor Calculus has significantly more compact expression trees. This is because Tensor Calculus uses index notation, which results in treating both the forward mode and the reverse mode in the same manner.

By contrast, Matrix Calculus hides the indices for notational convenience, and this often results in overly complicated automatic differentiation expression trees.

Consider the matrix multiplication $C = AB$. We have $\dot{C} = \dot{A}B + A\dot{B}$ for the forward mode and $\bar{A} = \bar{C}B^T, \bar{B} = A^T\bar{C}$ for the reverse mode. To perform the multiplications correctly, we have to be careful about the order of multiplication and the use of transposes. Notationally, this is a point of confusion for the machine learning practitioner, but computationally, this is an overhead for the program.

Here's another example, which is decidedly less trivial: $c = det(A)$. We have $\dot{c} = tr(inv(A)\dot{A})$ for the forward mode, and $\bar{A} = \bar{c}cinv(A)^T$ for the reverse mode. In this case, it is clearly not possible to use the same expression tree for both modes, given that they are composed of different operations.

In general, the way TensorFlow and other libraries (e.g. Mathematica, Maple, Sage, SimPy, ADOL-C, TAPENADE, TensorFlow, Theano, PyTorch, HIPS autograd) implement automatic differentiation results in different and inefficient expression trees for the forward and reverse mode. Tensor calculus conveniently avoids these problems by having commutativity in multiplication

as a result of its index notation. (Please read the actual paper to learn more about how this works.)

The authors tested their method of doing reverse-mode automatic differentiation, aka backpropagation, on three different problems and measured the amount of time it took to compute the Hessians.

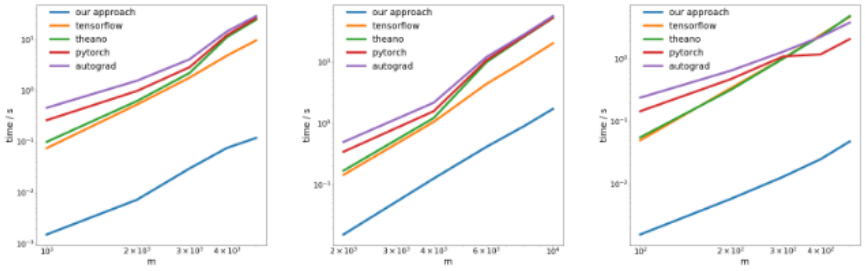


Figure 3: Log-log plot of the running times for evaluating the Hessian of the quadratic function (left), logistic regression (middle), matrix factorization (right) on the CPU. See the supplemental material for a table with the running times.

The first problem involves optimizing a quadratic function like $x^T Ax$. The second problem solves for logistic regression, while the third problem solves for matrix factorization.

On the CPU, their method was faster than popular automatic differentiation libraries like TensorFlow, Theano, PyTorch, and HIPS autograd by *two* orders of magnitude.

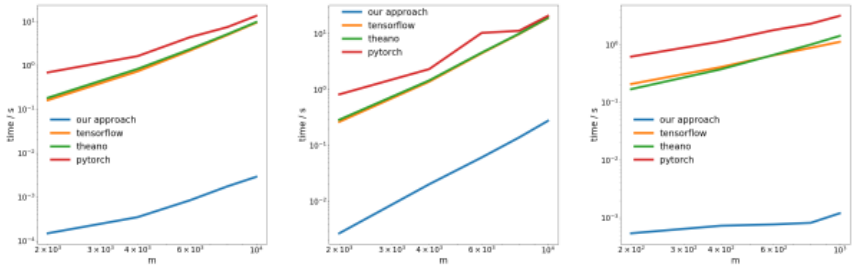


Figure 3: Log-log plot of the running times on the GPU for evaluating the Hessian for the quadratic function (left), logistic regression (middle), and matrix factorization (right).

On the GPU, they observed an even greater speedup, outperforming these libraries by a factor of *three* orders of magnitude.

Implication:

Computing derivatives for quadratic or higher functions with current deep learning libraries is more expensive than it needs to be. This includes computing general fourth order tensors like the Hessian (e.g. in MAML and second-order Newton optimization). Fortunately, quadratic functions are not common in deep learning. But they are common in classical machine learning - dual of an SVM, least squares regression, LASSO, Gaussian Processes, etc.

Myth 2: Image datasets are representative of real images found in the wild

We like to think that neural networks are now better than humans at the task of object recognition. This is not true. They might outperform humans on select image datasets like ImageNet, but given actual images found in the wild, they are most definitely not going to be better than a regular adult human at recognizing objects. This is because images found in current image datasets are not actually drawn from the same distribution as the set of all possible images naturally occurring in the wild.

In an old paper Torralba and Efros [2011], the authors proposed to examine dataset bias in twelve popular image datasets by observing if it is possible to train a classifier to identify the dataset a given image is selected from.

Figure 1. Name That Dataset: Given three images from twelve popular object recognition datasets, can you match the images with the dataset? (answer key below)

The chance of getting it right by random is $\frac{1}{12} \approx 8\%$, while their lab members performed at $> 75\%$.

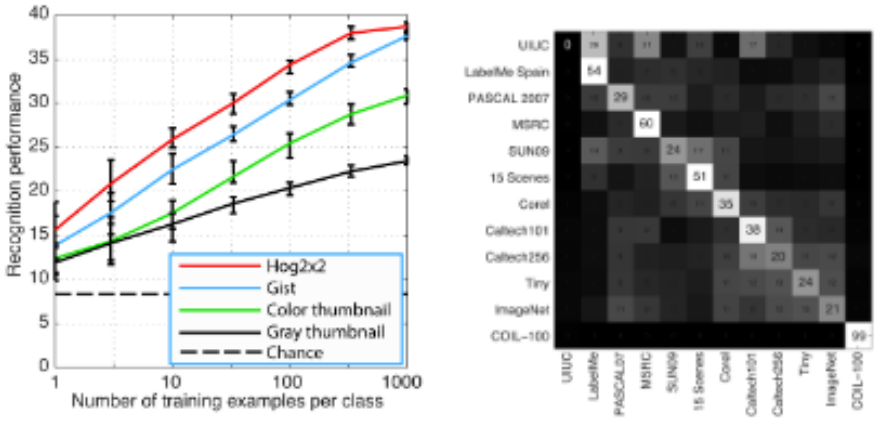


Figure 2. Computer plays *Name That Dataset*. Left: classification performance as a function of dataset size (log scale) for different descriptors (notice that performance does not appear to saturate). Right: confusion matrix.

They trained an SVM on HOG features, and found that their classifier performed at 39%, way above chance. If the same experiment was repeated today with a state of the art CNN, we will probably see a further increase in classifier performance.

If image datasets are truly representative of real images found in the wild, we ought to not be able to distinguish which dataset a given image originates from.

Figure 4. Most discriminative cars from 5 datasets

But there are biases in the data that make each dataset distinctive. For example, there are many race cars in the ImageNet dataset, which cannot be said to represent the "platonic" concept of a car in general.

Table 3. "Market Value" for a "car" sample across datasets

	SUN09 market	LabelMe market	PASCAL market	ImageNet market	Caltech101 market
1 SUN09 is worth	1 SUN09	0.91 LabelMe	0.72 pascal	0.41 ImageNet	0 Caltech
1 LabelMe is worth	0.41 SUN09	1 LabelMe	0.26 pascal	0.31 ImageNet	0 Caltech
1 pascal is worth	0.29 SUN09	0.50 LabelMe	1 pascal	0.88 ImageNet	0 Caltech
1 ImageNet is worth	0.17 SUN09	0.24 LabelMe	0.40 pascal	1 ImageNet	0 Caltech
1 Caltech101 is worth	0.18 SUN09	0.23 LabelMe	0 pascal	0.28 ImageNet	1 Caltech
Basket of Currencies	0.41 SUN09	0.58 LabelMe	0.48 pascal	0.58 ImageNet	0.20 Caltech

The authors further judged the value of a dataset by measuring how well a classifier trained on it performs on other datasets. By this metric, LabelMe and ImageNet are the least biased datasets, scoring 0.58 in a "basket of currencies." The values are all less than one, which means that training on a different dataset always results in lower test performance. In an ideal world without dataset bias, some of these values should be above one.

The authors pessimistically concluded:

> So, what is the value of current datasets when used to train algorithms that will be deployed in the real world? The answer that emerges can be summarized as: better than nothing, but not by

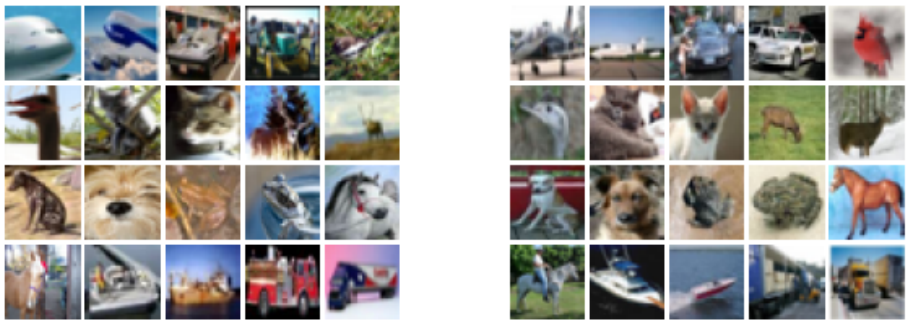
much

Myth 3: Machine Learning researchers do not use the test set for validation

In Machine Learning 101, we are taught to split a dataset into training, validation, and test sets. The performance of a model trained on the training set and evaluated on the validation set helps the machine learning practitioner tune his model to maximize its performance in real world usage. The test set should be held out until the practitioner is done with the tuning so as to provide an unbiased estimate of the model's actual performance in real world usage. If the practitioner "cheats" by using the test set in the training or validation process, he runs the risk of overfitting his model to biases inherent in the dataset that do not generalize beyond the dataset.

In the hyper-competitive world of machine learning research, new algorithms and models are often evaluated using their performance on the test set. Thus, there is little reason for researchers to write or submit papers that propose methods with inferior test performance. This effectively means that the machine learning research community, as a whole, is using the test set for validation.

What is the impact of this "cheating"?



(a) Test Set A (b) Test Set B

Figure 1: Class-balanced random draws from the new and original test sets.[1]

The authors of Recht et al. [2018] investigated this by creating a new test set for CIFAR-10. They did this by parsing images from the Tiny Images repository, as was done in the original dataset collection process.

They chose CIFAR-10 because it is one of the most widely used datasets in machine learning, being the second most popular dataset in NeurIPS 2017 (after MNIST). The dataset creation process for CIFAR-10 is also well-documented and transparent, with the large Tiny Images repository having sufficiently fine-grained labels that make it possible to replicate a new test set while minimizing distributional shift.

Table 1: Model accuracy on the original CIFAR-10 test set and the new test set, with the gap reported as the difference between the two accuracies. Δ Rank is the relative difference in the ranking from the original test set to the new test set. For example, ΔRank = −2 means a model dropped in the rankings by two positions on the new test set.

	Original Accuracy	New Accuracy	Gap	Δ Rank
shake_shake_64d_cutout [3, 4]	97.1 [96.8, 97.4]	93.0 [91.8, 94.0]	4.1	0
shake_shake_96d [4]	97.1 [96.7, 97.4]	91.9 [90.7, 93.1]	5.1	-2
shake_shake_64d [4]	97.0 [96.6, 97.3]	91.4 [90.1, 92.6]	5.6	-2
wide_resnet_28_10_cutout [3, 22]	97.0 [96.6, 97.3]	92.0 [90.7, 93.1]	5	+1
shake_drop [21]	96.9 [96.5, 97.2]	92.3 [91.0, 93.4]	4.6	+3
shake_shake_32d [4]	96.6 [96.2, 96.9]	89.8 [88.4, 91.1]	6.8	-2
darc [11]	96.6 [96.2, 96.9]	89.5 [88.1, 90.8]	7.1	-4
resnext_29_4x64d [20]	96.4 [96.0, 96.7]	89.6 [88.2, 90.9]	6.8	-2
pyramidnet_basic_110_270 [6]	96.3 [96.0, 96.7]	90.5 [89.1, 91.7]	5.9	+3
resnext_29_8x64d [20]	96.2 [95.8, 96.6]	90.0 [88.6, 91.2]	6.3	+3
wide_resnet_28_10 [22]	95.9 [95.5, 96.3]	89.7 [88.3, 91.0]	6.2	+2
pyramidnet_basic_110_84 [6]	95.7 [95.3, 96.1]	89.3 [87.8, 90.6]	6.5	0
densenet_BC_100_12 [10]	95.5 [95.1, 95.9]	87.6 [86.1, 89.0]	8	-2
neural_architecture_search [23]	95.4 [95.0, 95.8]	88.8 [87.4, 90.2]	6.6	+1
wide_resnet_tf [22]	95.0 [94.6, 95.4]	88.5 [87.0, 89.9]	6.5	+1
resnet_v2_bottleneck_164 [8]	94.2 [93.7, 94.6]	85.9 [84.3, 87.4]	8.3	-1
vgg16_keras [14, 18]	93.6 [93.1, 94.1]	85.3 [83.6, 86.8]	8.3	-1
resnet_basic_110 [7]	93.5 [93.0, 93.9]	85.2 [83.5, 86.7]	8.3	-1
resnet_v2_basic_110 [8]	93.4 [92.9, 93.9]	86.5 [84.9, 88.0]	6.9	+3
resnet_basic_56 [7]	93.3 [92.8, 93.8]	85.0 [83.3, 86.5]	8.3	0
resnet_basic_44 [7]	93.0 [92.5, 93.5]	84.2 [82.6, 85.8]	8.8	-3
vgg_15_BN_64 [14, 18]	93.0 [92.5, 93.5]	84.9 [83.2, 86.4]	8.1	+1
resnet_preact_tf [7]	92.7 [92.2, 93.2]	84.4 [82.7, 85.9]	8.3	0
resnet_basic_32 [7]	92.5 [92.0, 93.0]	84.9 [83.2, 86.4]	7.7	+3
cudaconvnet [13]	88.5 [87.9, 89.2]	77.5 [75.7, 79.3]	11	0
random_features_256k_aug [2]	85.6 [84.9, 86.3]	73.1 [71.1, 75.1]	12	0
random_features_32k_aug [2]	85.0 [84.3, 85.7]	71.9 [69.9, 73.9]	13	0
random_features_256k [2]	84.2 [83.5, 84.9]	69.9 [67.8, 71.9]	14	0
random_features_32k [2]	83.3 [82.6, 84.0]	67.9 [65.9, 70.0]	15	-1
alexnet_tf	82.0 [81.2, 82.7]	68.9 [66.8, 70.9]	13	+1

They found that across a wide range of different neural network models, there was a significant drop in accuracy ($4\% - 15\%$) from the old test set to the new test set. However, the relative ranking of each model's performance remained fairly stable.

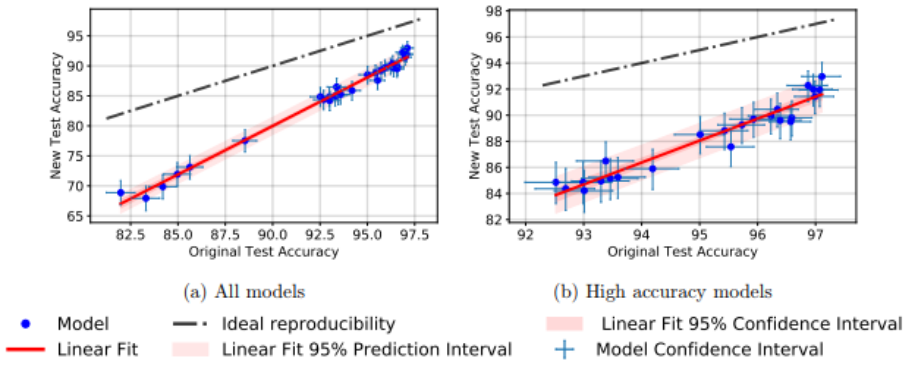


Figure 2: Model accuracy on new test set vs. model accuracy on original test set.

In general, the higher performing models experienced a smaller drop in accuracy compared to the lower performing models. This is heartening, because it indicates that the loss in generalization caused by the "cheating," at least in the case of CIFAR-10, becomes more muted as the research community invents better machine learning models and methods.

Myth 4: Every datapoint is used in training a neural network

Conventional wisdom says that data is the new oil and the more data we have, the better we can train our sample-inefficient and overparametrized deep learning models.

In Toneva et al. [2018], the authors demonstrate significant redundancy in several common small image datasets. Shockingly, 30% of the datapoints in CIFAR-10 can be removed, without changing test accuracy by much.

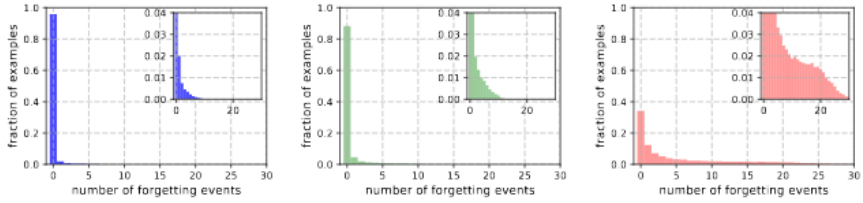


Figure 1: Histograms of forgetting events on (from left to right) *MNIST*, *permutedMNIST* and *CIFAR-10*. Insets show the zoomed-in y-axis.

A forgetting event happens when the neural network makes a misclassification at time $t+1$, having already made an accurate classification at time t, where we consider the flow of time to be the number of SGD updates made to the network. To make the tracking of forgetting events tractable, the authors run their neural network over only the examples in the mini-batch every time an SGD update is made, rather than over every single example in the dataset. Examples

that do not undergo a forgetting event are called *unforgettable* examples.

They find that 91.7% of MNIST, 75.3% of permutedMNIST, 31.3% of CIFAR-10, and 7.62% of CIFAR-100 comprise of unforgettable examples. This makes intuitive sense, since an increase in the diversity and complexity of an image dataset should cause the neural network to forget more examples.

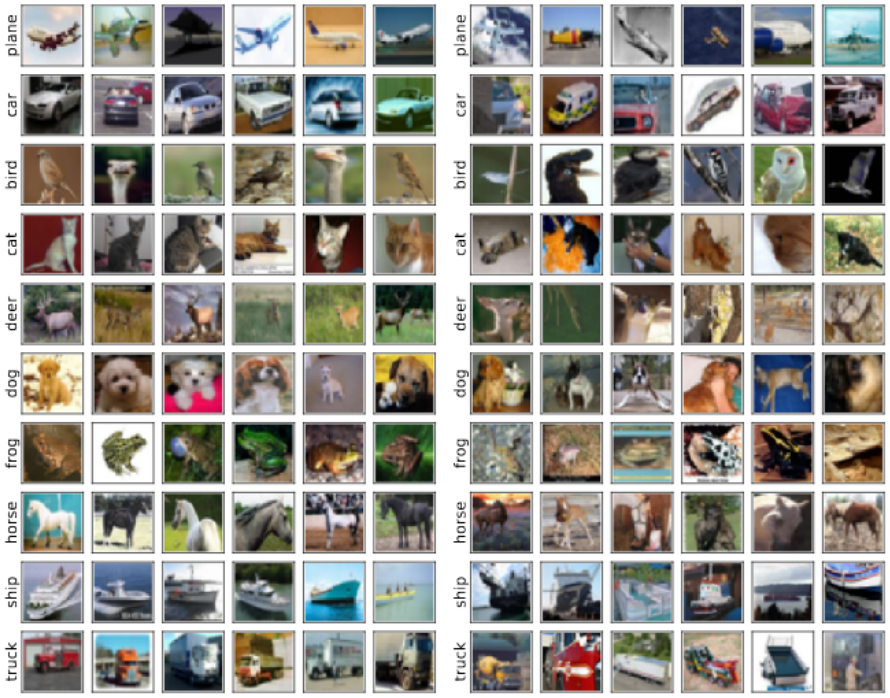


Figure 15: Additional pictures of the most unforgettable (*Left*) and forgettable examples (*Right*) of every *CIFAR-10* class, when examples are sorted by number of forgetting events (ties are broken randomly). Forgettable examples seem to exhibit peculiar or uncommon features.

Forgettable examples seem to display more uncommon and peculiar features than unforgettable examples. The authors liken them to support vectors in SVM, because they seem to demarcate the contours of the decision boundary.

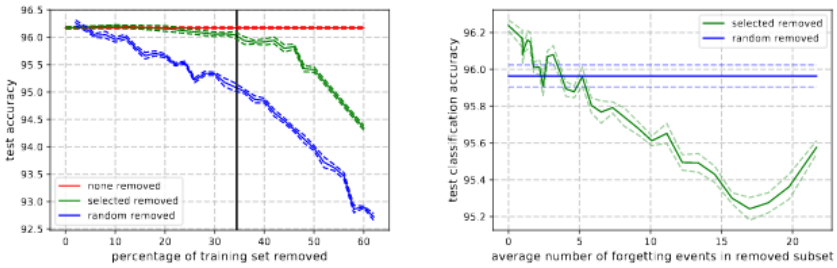


Figure 5: *Left* Generalization performance on *CIFAR-10* of ResNet18 where increasingly larger subsets of the training set are removed (mean +/- std error of 5 seeds). When the removed examples are selected at random, performance drops very fast. Selecting the examples according to our ordering can reduce the training set significantly without affecting generalization. The vertical line indicates the point at which all unforgettable examples are removed from the training set. *Right* Difference in generalization performance when contiguous chunks of 5000 increasingly forgotten examples are removed from the training set. Most important examples tend to be those that are forgotten the most.

Unforgettable examples, by contrast, encode mostly redundant information. If we sort the examples by their unforgettability, we can compress the dataset by removing the most unforgettable examples.

On CIFAR-10, 30% of the dataset can be removed without affecting test accuracy, while a 35% removal causes a trivial 0.2% dip in test accuracy. If this 30% was selected by random instead of chosen by unforgettability, then its removal will result in a significant loss of around 1% in test accuracy.

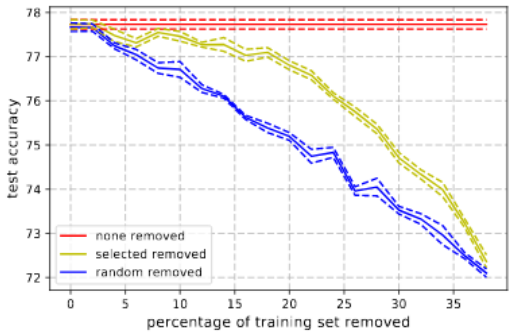


Figure 18: Generalization performance on *CIFAR-100* of ResNet18 where increasingly larger subsets of the training set are removed (mean +/- std error of 5 seeds). When the removed examples are selected at random, performance drops faster. Selecting the examples according to our ordering reduces the training set without affecting generalization.

Similarly, on CIFAR-100, 8% of the dataset can be removed without affecting test accuracy.

These findings show that there is significant data redundancy in neural network training, much like in SVM training where the non-support vectors can be taken away without affecting the decisions of the model.

Implication:

If we can determine which examples are unforgettable before the start of training, then we can save space by removing those examples and save time by not training the neural network on them.

Myth 5: We need (batch) normalization to train very deep residual networks

It had been believed for a very long time that "training a deep network to directly optimize only the supervised objective of interest (for example the log probability of correct classification) by gradient descent, starting from random initialized parameters, does not work very well." [Vincent et al., 2010]

Since then, a host of clever random initialization methods, activation functions, optimization techniques, and other architectural innovations like residual connections [He et al., 2016], has made it easier to train deep neural networks with gradient descent.

But the real breakthrough came from the introduction of batch normalization [Ioffe and Szegedy, 2015] (and other subsequent normalization techniques), which constrained the size of activations at every layer of a deep network to mitigate the vanishing and exploding gradients problem.

In a recent paper Zhang et al. [2019], it was shown remarkably that it is actually possible to train a $10,000$-layer deep network using vanilla SGD, without resorting to any normalization.

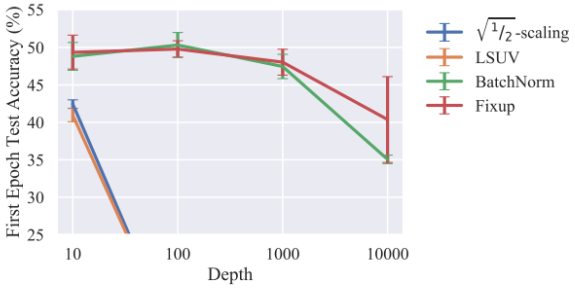


Figure 3: Depth of residual networks versus test accuracy at the first epoch for various methods on CIFAR-10 with the default BatchNorm learning rate. We observe that Fixup is able to train very deep networks with the same learning rate as batch normalization. (Higher is better.)

The authors compared training a residual network at varying depths for one epoch on CIFAR-10, and found that while standard initialization methods failed for 100 layers, both Fixup and batch normalization succeeded for $10,000$ layers.

Fixup initialization (or: How to train a deep residual network without normalization)

1. Initialize the classification layer and the last layer of each residual branch to 0.
2. Initialize every other layer using a standard method (e.g., He et al. (2015)), and scale only the weight layers inside residual branches by $L^{-\frac{1}{2m-2}}$.
3. Add a scalar multiplier (initialized at 1) in every branch and a scalar bias (initialized at 0) before each convolution, linear, and element-wise activation layer.

They did a theoretical analysis to show that "the gradient norm of certain layers is in expectation lower bounded by a quantity that increases indefinitely with the network depth," i.e. the exploding gradients problem.

To prevent this, the key idea in Fixup is to scale the weights in the m layers for each of L residual branches by a factor that depends on m and L.

Dataset	ResNet-110	Normalization	Large η	Test Error (%)
CIFAR-10	w/ BatchNorm (He et al., 2016)	✓	✓	6.61
	w/ Xavier Init (Shang et al., 2017)	✗	✗	7.78
	w/ Fixup-init	✗	✓	7.24

Table 1: Results on CIFAR-10 with ResNet-110 (mean/median of 5 runs; lower is better).

Fixup enabled the training of a deep residual network with 110 layers on CIFAR-10 with a large learning rate, at comparable test performance with the same network architecture that had batch normalization.

Model	Method	Normalization	Test Error (%)
ResNet-50	BatchNorm (Goyal et al., 2017)	✓	23.6
	BatchNorm + Mixup (Zhang et al., 2017)		**23.3**
	GroupNorm + Mixup		23.9
	Xavier Init (Shang et al., 2017)	✗	31.5
	Fixup-init		27.6
	Fixup-init + Mixup		**24.0**
ResNet-101	BatchNorm (Zhang et al., 2017)	✓	22.0
	BatchNorm + Mixup (Zhang et al., 2017)		**20.8**
	GroupNorm + Mixup		21.4
	Fixup-init + Mixup	✗	21.4

Table 2: ImageNet test results using the ResNet architecture. (Lower is better.)

Dataset	Model	Normalization	BLEU
IWSLT DE-EN	(Deng et al., 2018)	✓	33.1
	LayerNorm		34.2
	Fixup-init	✗	**34.5**
WMT EN-DE	(Vaswani et al., 2017)	✓	28.4
	LayerNorm (Ott et al., 2018)		**29.3**
	Fixup-init	✗	**29.3**

Table 3: Comparing Fixup vs. LayerNorm for machine translation tasks. (Higher is better.)

The authors also further showed comparable test results using a Fixuped network without any normalization on the ImageNet dataset and English-German machine translation tasks.

Myth 6: Attention > Convolution

There is an idea gaining currency in the machine learning community that attention mechanisms are a superior alternative to convolutions now [Vaswani et al., 2017]. Importantly, Vaswani et al. [2017] noted that "the computational cost of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer."

Even state-of-the-art GANS find self-attention superior to standard convolutions in its ability to model long-range, multi-scale dependencies [Zhang et al., 2018].

The authors of Wu et al. [2019] question the parameter efficiency and efficacy of self-attention in modelling long-range dependencies, and propose new variants of convolutions, partially inspired by self-attention, that are more parameter-efficient.

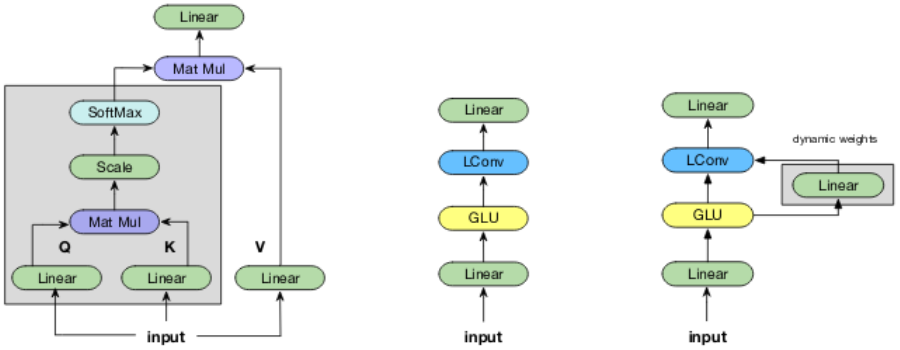


Figure 2: Illustration of self-attention, lightweight convolutions and dynamic convolutions.

Lightweight convolutions are depthwise-separable, softmax-normalized across the temporal dimension, shares weights across the channel dimension, and reuses the same weights at every time step (like RNNs). *Dynamic* convolutions are lightweight convolutions that use different weights at every time step.

These tricks make lightweight and dynamic convolutions several orders of magnitude more efficient that standard non-separable convolutions.

Model	Param (En-De)	WMT En-De	WMT En-Fr
Gehring et al. (2017)	216M	25.2	40.5
Vaswani et al. (2017)	213M	28.4	41.0
Ahmed et al. (2017)	213M	28.9	41.4
Chen et al. (2018)	379M	28.5	41.0
Shaw et al. (2018)	-	29.2	41.5
Ott et al. (2018)	210M	29.3	**43.2**
LightConv	202M	28.9	43.1
DynamicConv	213M	**29.7**	**43.2**

Table 1: Machine translation accuracy in terms of BLEU for WMT En-De and WMT En-Fr on newstest2014.

Model	Param (Zh-En)	IWSLT	WMT Zh-En
Deng et al. (2018)	-	33.1	-
Hassan et al. (2018)	-	-	24.2
Self-attention baseline	292M	34.4	23.8
LightConv	285M	34.8	24.3
DynamicConv	296M	**35.2**	**24.4**

Table 2: Machine translation accuracy in terms of BLEU on IWSLT and WMT Zh-En.

Model	Param	Valid	Test
2-layer LSTM-8192-1024 (Józefowicz et al., 2016)	–	–	30.6
Gated Convolutional Model (Dauphin et al., 2017)	428M	–	31.9
Mixture of Experts (Shazeer et al., 2017)	4371M †	–	28.0
Self-attention baseline	331M	26.67	26.73
DynamicConv	339M	26.60	**26.67**

Table 4: Language modeling results on the Google Billion Word test set.
†does not include embedding and softmax layers

Model	Param	Rouge-1	Rouge-2	Rouge-l
LSTM (Paulus et al., 2017)	-	38.30	14.81	35.49
CNN (Fan et al., 2017)	-	39.06	15.38	35.77
Self-attention baseline	90M	39.26	15.98	36.35
LightConv	86M	39.52	15.97	36.51
DynamicConv	87M	**39.84**	**16.25**	**36.73**
RL (Celikyilmaz et al., 2018)	-	**41.69**	**19.47**	**37.92**

Table 5: Results on CNN-DailyMail summarization. We compare to likelihood trained approaches except for Celikyilmaz et al. (2018).

The authors show that these new convolutions match or exceed the self-attention baselines in machine translation, language modelling, and abstractive summarization tasks while using comparable or less number of parameters.

Myth 7: Saliency maps are robust ways to interpret neural networks.

While neural networks are commonly believed to be black boxes, there have been many, many attempts made to interpret them. Saliency maps, or other similar methods that assign importance scores to features or training examples, are the most popular form of interpretation.

It is tempting to be able to conclude that the reason why a given image is classified a certain way is due to particular parts of the image that are salient to the neural network's decision in making the classification. There are several ways to compute this saliency map, often making use of a neural network's activations on a given image and the gradients that flow through the network.

In Ghorbani et al. [2017], the authors show that they can introduce an imperceptible perturbation to a given image to distort its saliency map.

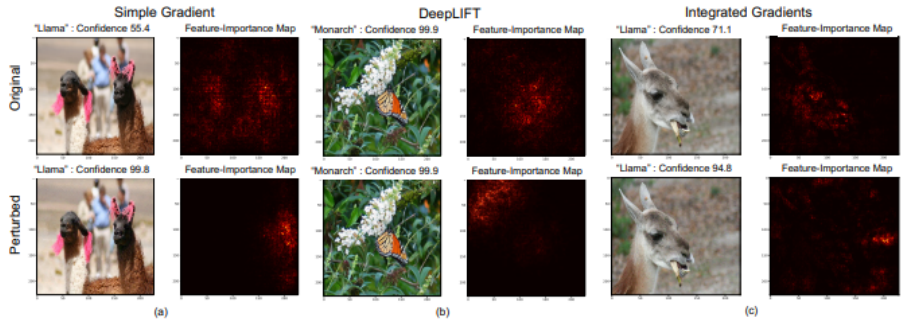


Figure 1: **Adversarial attack against feature-importance maps.** We generate feature-importance scores, also called saliency maps, using three popular interpretation methods: (a) simple gradients, (b) DeepLIFT, and (c) integrated gradients. The **top row** shows the the original images and their saliency maps and the **bottom row** shows the perturbed images (using the center attack with $\epsilon = 8$, as described in Section 3) and corresponding saliency maps. In all three images, the predicted label does not change from the perturbation; however, the saliency maps of the perturbed images shifts dramatically to features that would not be considered salient by human perception.

A monarch butterfly is thus classified as a monarch butterfly, not on account of the patterns on its wings, but because of some unimportant green leaves in the background.

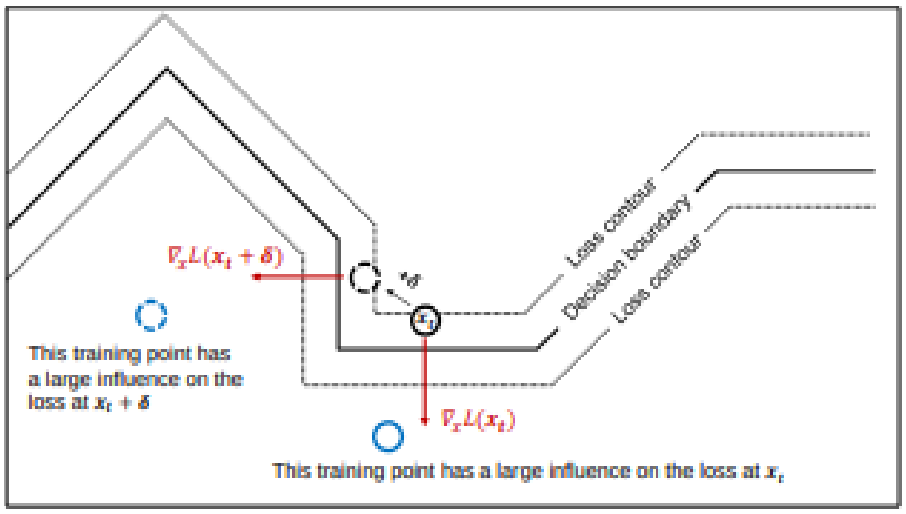


Figure 2: **Intuition for why interpretation is fragile.** Con-

High-dimensional images often lie close to the decision boundaries constructed by deep neural networks, hence their susceptibility to adversarial attacks. While adversarial attacks shift images past a decision boundary, adversarial interpretation attacks shift them along the contour of the decision boundary, while still remaining within the same decision territory.

Algorithm 1 Iterative feature importance Attacks

Input: test image $\boldsymbol{x}_t$, maximum norm of perturbation ϵ, normalized feature importance function $\boldsymbol{I}(\cdot)$, number of iterations P, step size α

Define a dissimilarity function D to measure the change between interpretations of two images:

$$D(\boldsymbol{x}_t, \boldsymbol{x}) = \begin{cases} -\sum_{i \in B} \boldsymbol{I}(\boldsymbol{x})_i & \text{for } \textbf{top-k} \text{ attack} \\ \sum_{i \in \mathcal{A}} \boldsymbol{I}(\boldsymbol{x})_i & \text{for } \textbf{targeted} \text{ attack} \\ ||\boldsymbol{C}(\boldsymbol{x}) - \boldsymbol{C}(\boldsymbol{x}_t)||_2 & \text{for } \textbf{mass-center} \text{ attac} \end{cases}$$

where B is the set of the k largest dimensions of $\boldsymbol{I}(\boldsymbol{x}_t)$, $\mathcal{A}$ is the target region of the input image in targeted attack, and $\boldsymbol{C}(\cdot)$ is the center of feature importance mass[a].

Initialize $\boldsymbol{x}^0 = \boldsymbol{x}_t$

for $p \in \{1, \ldots, P\}$ **do**

Perturb the test image in the direction of signed gradient[b] of the dissimilarity function:

$$\boldsymbol{x}^p = \boldsymbol{x}^{p-1} + \alpha \cdot \text{sign}(\nabla_{\boldsymbol{x}} D(\boldsymbol{x}_t, \boldsymbol{x}^{p-1}))$$

If needed, clip the perturbed input to satisfy the norm constraint: $||\boldsymbol{x}^p - \boldsymbol{x}_t||_\infty \leq \epsilon$

end for

Among $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^P\}$, return the element with the largest value for the dissimilarity function and the same prediction as the original test image.

[a] The center of mass is defined for a $W \times H$ image as: $\boldsymbol{C}(\boldsymbol{x}) = \sum_{i \in \{1,\ldots,W\}} \sum_{j \in \{1,\ldots,H\}} \boldsymbol{I}(\boldsymbol{x})_{i,j} [i, j]^T$

[b] In ReLU networks, this gradient is 0. To attack interpretability in such networks, we replace the ReLU activation with its smooth approximation (softplus) when calculating the gradient and generate the perturbed image using this approximation. The perturbed images that result are effective adversarial attacks against the original ReLU network, as discussed in Section 4.

The basic method employed by the authors to do this is a modification of Goodfellow et al. [2014]'s fast gradient sign method, which was one of the first efficient adversarial attacks introduced. This suggests that other more recent and sophisticated adversarial attacks can also be used to attack neural network interpretations.

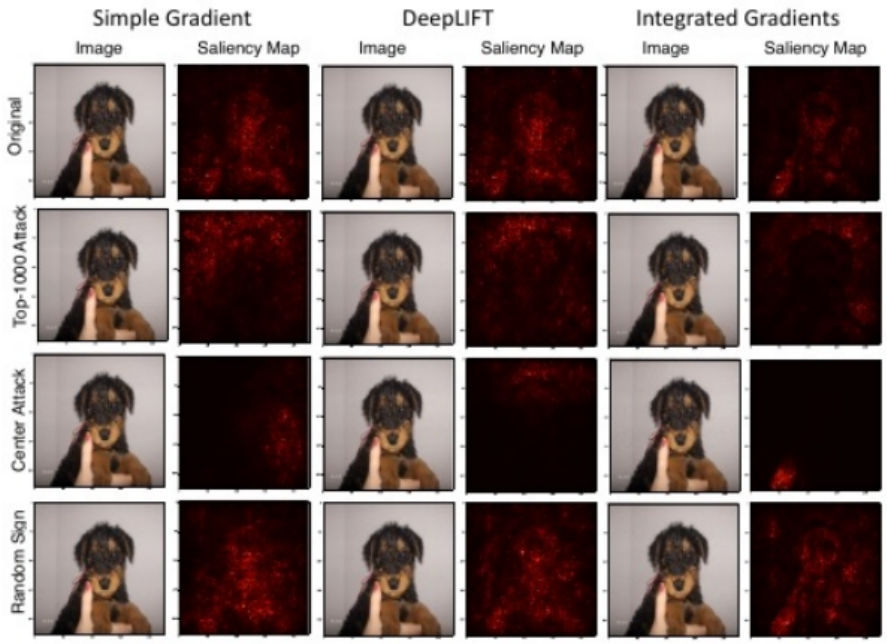


Figure 7: All of the images are classified as a *airedale*.

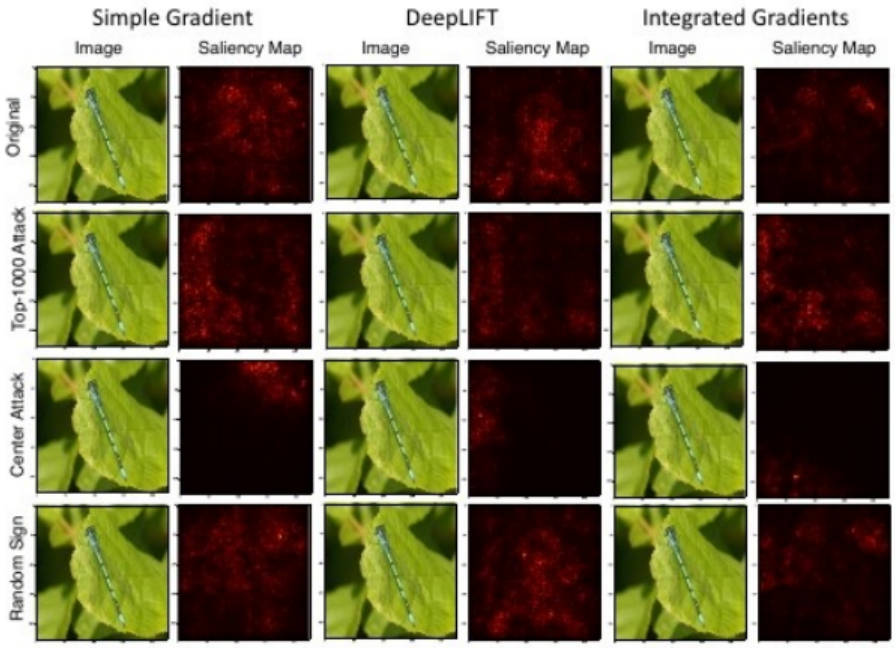


Figure 8: All of the images are classified as a *damselfly*.

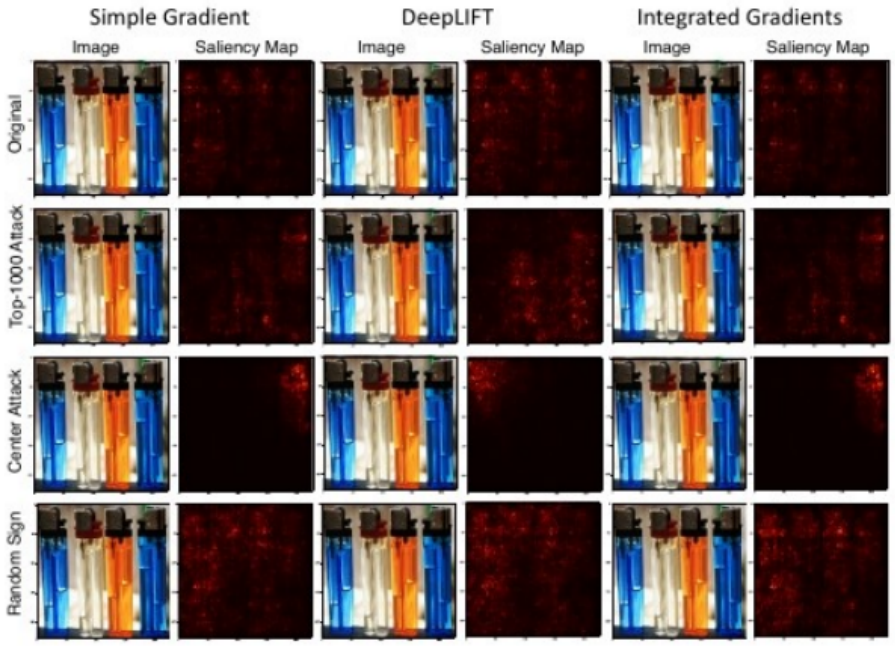


Figure 9: All of the images are classified as a *lighter*.

Implication:

As deep learning becomes more and more ubiquitous in high stakes applications like medical imaging, it is important to be careful of how we interpret decisions made by neural networks. For example, while it would be nice to have a CNN identify a spot on an MRI image as a malignant cancer-causing tumor, these results should not be trusted if they are based on fragile interpretation methods.

References

Sören Laue, Matthias Mitterreiter, and Joachim Giesen. Computing higher order derivatives of matrix and tensor expressions. In *Advances in Neural Information Processing Systems*, pages 2755–2764, 2018.

Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. 2011.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*, 2018.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forget-

ting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. *arXiv preprint arXiv:1710.10547*, 2017.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.