Distributional Generalization and Local Elasticity

Presenter: Zhanyu Wang

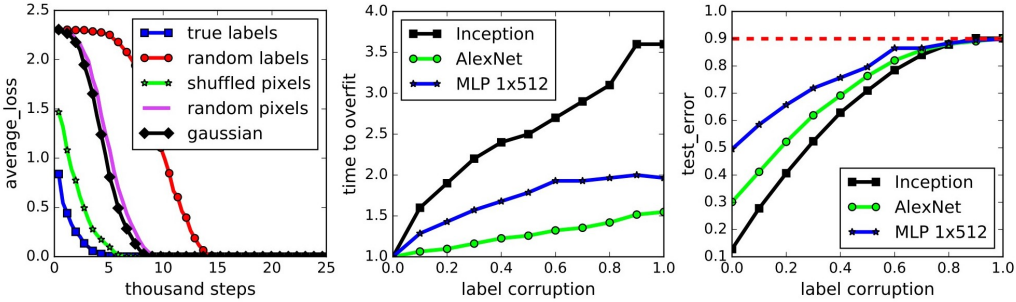
November 1, 2020

Understanding deep learning requires rethinking generalization[1]

- Conventional wisdom: small generalization error either by properties of the model family, or by the regularization techniques used during fitting.
- Experiments show that deep neural networks easily fit random labels.
  - Even optimization on random labels remains easy. In fact, training time increases only by a small constant factor compared with training on the true labels.
  - Randomizing labels is solely a data transformation, leaving all other properties of the learning problem unchanged.
  - Neural networks are able to capture the remaining signal in the data, while at the same time fit the noisy part using brute-force.
- Explicit regularization may improve generalization performance, but is neither necessary nor by itself sufficient for controlling generalization error. (Without weight decay, dropout, or data augmentation, it can still generalize.)

[1] ICLR2017 https://arxiv.org/pdf/1611.03530.pdf

Fitting random label is possible and not hard



(a) learning curves (b) convergence slowdown (c) generalization error growth

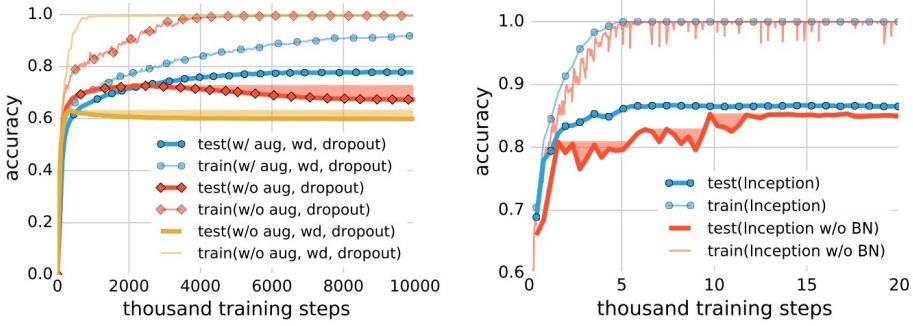
Figure 1: Fitting random labels and random pixels on CIFAR10. (a) shows the training loss of various experiment settings decaying with the training steps. (b) shows the relative convergence time with different label corruption ratio. (c) shows the test error (also the generalization error since training error is 0) under different label corruptions.

Regularizers can help but are not crucial

Table 1: The training and test accuracy (in percentage) of various models on the CIFAR10 dataset. Performance with and without data augmentation and weight decay are compared. The results of fitting random labels are also included.

model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48

Early stopping is needed when other regularizers are absent



(a) Inception on ImageNet (b) Inception on CIFAR10

Figure 2: Effects of implicit regularizers on generalization performance. `aug` is data augmentation, `wd` is weight decay, `BN` is batch normalization. The shaded areas are the cumulative best test accuracy, as an indicator of potential performance gain of early stopping. (a) early stopping could potentially improve generalization when other regularizers are absent. (b) early stopping is not necessarily helpful on CIFAR10, but batch normalization stablize the training process and improves generalization.

Fitting random labels is still possible under regularization

Table 2: The top-1 and top-5 accuracy (in percentage) of the Inception v3 model on the ImageNet dataset. We compare the training and test accuracy with various regularization turned on and off, for both true labels and random labels. The original reported top-5 accuracy of the Alexnet on ILSVRC 2012 is also listed for reference. The numbers in parentheses are the best test accuracy during training, as a reference for potential performance gain of early stopping.

data aug	dropout	weight decay	top-1 train	top-5 train	top-1 test	top-5 test
ImageNet 1000 classes with the original labels						
yes	yes	yes	92.18	99.21	77.84	93.92
yes	no	no	92.33	99.17	72.95	90.43
no	no	yes	90.60	100.0	67.18 (72.57)	86.44 (91.31)
no	no	no	99.53	100.0	59.80 (63.16)	80.38 (84.49)
Alexnet (Krizhevsky et al., 2012)			-	-	-	83.6
ImageNet 1000 classes with random labels						
no	yes	yes	91.18	97.95	0.09	0.49
no	no	yes	87.81	96.15	0.12	0.50
no	no	no	95.20	99.14	0.11	0.56

Challenge for traditional approaches on reasoning about generalization

Rademacher complexity and VC-dimension

- The empirical Rademacher complexity of a hypothesis class $\mathcal{H}$ on a dataset $\{x_1, \ldots, x_n\}$ is defined as

$$\widehat{\mathcal{R}}_n(\mathcal{H}) = \mathbb{E}_\sigma \left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right].$$

  where $\sigma_1, \ldots, \sigma_n \in -1, +1$ are i.i.d. uniform r.v.s.
- It measures ability of $\mathcal{H}$ to fit random ± 1 binary label assignments.
- Since many neural networks fit the training set with random labels perfectly, we expect that $\widehat{\mathcal{R}}_n(\mathcal{H}) \approx 1$ for the corresponding model class $\mathcal{H}$.

Uniform stability

- Uniform stability of an algorithm A measures how sensitive the algorithm is to the replacement of a single example.
- However, it is solely a property of the algorithm, which does not take into account specifics of the data or the distribution of the labels.

Classical statistics bounds [2]

C.1 VC-Dimension Based Measures

We start by restating the theorem in (Bartlett et al., 2019) which provides an upper bound on the VC-dimension of any piece-wise linear network.

Theorem 1 (Bartlett et al. (2019)) *Let $\mathcal{F}$ be the class of feed-forward networks with a fixed computation graph of depth d and ReLU activations. Let a_i and q_i be the number of activations and parameters in layer i. Then VC-dimension of $\mathcal{F}$ can be bounded as follows:*

$$VC(\mathcal{F}) \leq d + \left(\sum_{i=1}^{d} (d - i + 1) q_i \right) \log_2 \left(8e \sum_{i=1}^{d} i a_i \log_2 \left(4e \sum_{j=1}^{d} j a_j \right) \right)$$

Theorem 2 *Given a convolutional network f, for any $\delta > 0$, with probability $1 - \delta$ over the the training set:*

$$L \leq \hat{L} + 4000 \sqrt{\frac{d \log_2 (6dn)^3 \sum_{i=1}^{d} k_i^2 c_i c_{i-1}}{m}} + \sqrt{\frac{\log(1/\delta)}{m}} \tag{16}$$

[2] ICLR2020 https://arxiv.org/pdf/1912.02178.pdf

Classical statistics bounds

C.2 (Norm & Margin)-Based Measures

Several generalization bounds have been proved for neural networks using margin and norm notions. In this section, we go over several such measures. For fully connected networks, Bartlett and Mendelson (2002) have shown a bound based on product of $\ell_{1,\infty}$ norm of the layer weights times a 2^d factor where $\ell_{1,\infty}$ is the maximum over hidden units of the ℓ_2 norm of the incoming weights to the hidden unit. Neyshabur et al. (2015b) proved a bound based on product of Frobenius norms of the layer weights times a 2^d factor and Golowich et al. (2017) was able to improve the factor to $\sqrt{d}$. Bartlett et al. (2017) proved a bound based on product of spectral norm of the layer weights times sum over layers of ratio of Frobenius norm to spectral norm of the layer weights and Neyshabur et al. (2018a) showed a similar bound can be achieved in a simpler way using PAC-bayesian framework.

Spectral Norm Unfortunately, none of the above founds are directly applicable to convolutional networks. Pitas et al. (2017) built on Neyshabur et al. (2018a) and extended the bound on the spectral norm to convolutional networks. The bound is very similar to the one for fully connected networks by Bartlett et al. (2017). We next restate their generalization bound for convolutional networks including the constants.

Theorem 3 (Pitas et al. (2017)) *Let B an upper bound on the ℓ_2 norm of any point in the input domain. For any $B, \gamma, \delta > 0$, the following bound holds with probability $1-\delta$ over the training set:*

$$L \leq \hat{L}_\gamma + \sqrt{\frac{\left(84B\sum_{i=1}^{d} k_i\sqrt{c_i} + \sqrt{\ln(4n^2d)}\right)^2 \prod_{i=1}^{d} \|\mathbf{W}_i\|_2^2 \sum_{j=1}^{d} \frac{\|\mathbf{W}_j - \mathbf{W}_j^0\|_F^2}{\|\mathbf{W}_j\|_2^2} + \ln(\frac{m}{\delta})}{\gamma^2 m}} \tag{24}$$

Classical statistics bounds

C.3 Flatness-based Measures

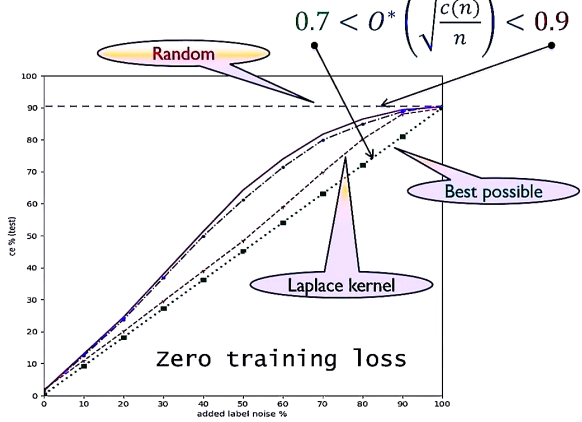
PAC-Bayesian framework (McAllester, 1999) allows us to study flatness of a solution and connect it to generalization. Given a prior P is is chosen before observing the training set and a posterior Q which is a distribution on the solutions of the learning algorithm (and hence depends on the training set), we can bound the expected generalization error of solutions generated from Q with high probability based on the KL divergence of P and Q. The next theorem states a simplified version of PAC-Bayesian bounds.

Theorem 4 *For any $\delta > 0$, distribution D, prior P, with probability $1-\delta$ over the training set, for any posterior Q the following bound holds:*

$$\mathbb{E}_{\mathbf{v}\sim Q}\left[L(f_{\mathbf{v}})\right] \leq \mathbb{E}_{\mathbf{w}\sim Q}\left[\hat{L}(f_{\mathbf{v}})\right] + \sqrt{\frac{\mathrm{KL}(Q||P) + \log\left(\frac{m}{\delta}\right)}{2(m-1)}} \tag{46}$$

From Classical Statistics to Modern Machine Learning [3]

What kind of generalization bound could work here?



[3] https://www.youtube.com/watch?v=OBCciGnOJVs

From Classical Statistics to Modern Machine Learning

- VC-dimension/Rademacher complexity/covering/margin bounds.
  - Cannot deal with interpolated classifiers when Bayes risk is non-zero.
  - Generalization gap cannot be bound when empirical risk is zero.
- Regularization-type analyses (Tikhonov, early stopping/SGD, etc.)
  - Diverge as $\lambda \to 0$ for fixed n.
- Algorithmic stability.
  - Does not apply when empirical risk is zero, expected risk nonzero.

WYSIWYG bounds:

training loss $= 0$

$\approx$

expected loss

- Classical smoothing methods (i.e., Nadaraya–Watson).
  - Most classical analyses do not support interpolation.
  - But 1-NN! (Also Hilbert regression Scheme, [Devroye, et al. 98])

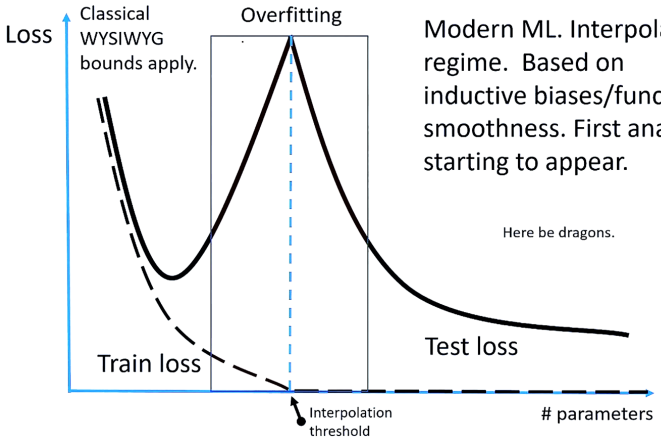
Oracle bounds

expected loss

$\approx$

optimal loss

From Classical Statistics to Modern Machine Learning



Modern ML. Interpolation regime. Based on inductive biases/functional smoothness. First analyses starting to appear.

Here be dragons.

Label noise in one class (this class is unknown by the given label)[4]

Experiment 1. *Consider a binary classification version of CIFAR-10, where CIFAR-10 images x have binary labels* `Animal/Object`*. Take 50K samples from this distribution as a train set, but apply the following label noise: flip the label of cats to* `Object` *with probability 30%. Now train a WideResNet f to 0 train error on this train set. How does the trained classifier behave on test samples? Some potential options are:*

1. The test error is uniformly small across all CIFAR-10 classes, since there is only 3% overall label noise in the train set.
2. The test error is moderate, and "spread" across all animal classes. After all, the classifier is not explicitly told what a cat or a dog is, just that they are all animals.
3. The test error is localized: the classifier misclassifies roughly 30% of test cats as "objects", but all other types of animals are largely unaffected.

[4] https://arxiv.org/pdf/2009.08092.pdf (also check 'Related Work' in their paper)
https://media.mis.mpg.de/mml/2020-08-21/

Label noise in one class only affects this class

In fact, reality is closest to option (3), for essentially any good architecture trained to interpolation. Figure 1 shows the results of this experiment with a WideResNet [Zagoruyko and Komodakis, 2016]. The left panel shows the joint density of (x, y) of inputs x and labels $y \in \{\texttt{Object/Animal}\}$ on the train set. Since the classifier f is interpolating, this joint distribution is identical to the classifier's outputs $(x, f(x))$ on the train set. The right panel shows the joint density of $(x, f(x))$ of the classifier's predictions on *test inputs* x.

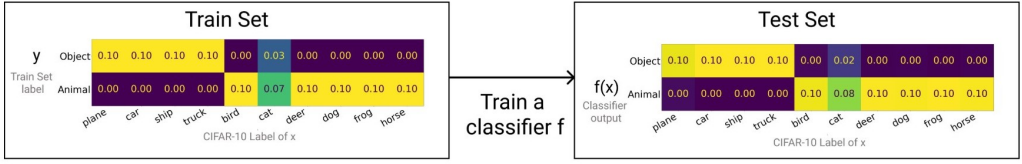


Figure 1: The setup and result of Experiment 1. The CIFAR-10 train set is labeled as either Animals or Objects, with label noise affecting only cats. A WideResNet-28-10 is then trained to 0 train error on this train set, and evaluated on the test set. The joint distribution of $(x, f(x))$ on the train set is close to $(x, f(x))$ on the test set. Full experimental details in Appendix D.2.

Label noise in one class only affects this class

In fact, reality is closest to option (3), for essentially any good architecture trained to interpolation. Figure 1 shows the results of this experiment with a WideResNet [Zagoruyko and Komodakis, 2016]. The left panel shows the joint density of (x, y) of inputs x and labels $y \in \{\texttt{Object/Animal}\}$ on the train set. Since the classifier f is interpolating, this joint distribution is identical to the classifier's outputs $(x, f(x))$ on the train set. The right panel shows the joint density of $(x, f(x))$ of the classifier's predictions on *test inputs* x.

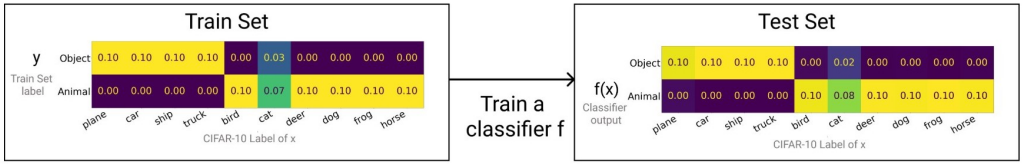
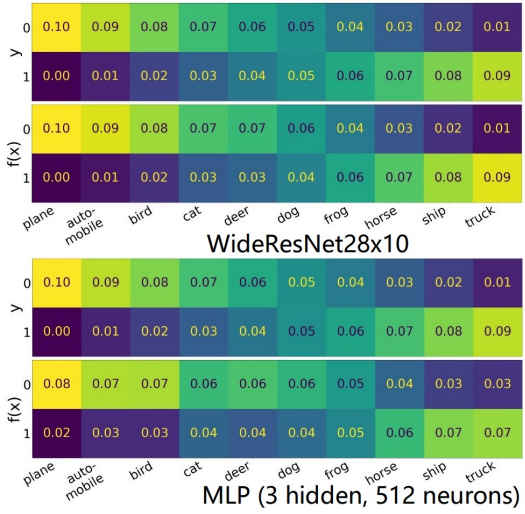


Figure 1: The setup and result of Experiment 1. The CIFAR-10 train set is labeled as either Animals or Objects, with label noise affecting only cats. A WideResNet-28-10 is then trained to 0 train error on this train set, and evaluated on the test set. The joint distribution of $(x, f(x))$ on the train set is close to $(x, f(x))$ on the test set. Full experimental details in Appendix D.2.

Label noise in each class affects the class locally



Distributional Generalization on model-distinguishable features

Classical Generalization: *Let f be a classifier trained on a set of samples (TrainSet). Then f generalizes if:*

$$\mathop{\mathbb{E}}_{\substack{x\sim TrainSet\\ \widehat{y}\leftarrow f(x)}}[\mathbb{1}\{\widehat{y}\neq y(x)\}] \approx \mathop{\mathbb{E}}_{\substack{x\sim TestSet\\ \widehat{y}\leftarrow f(x)}}[\mathbb{1}\{\widehat{y}\neq y(x)\}] \tag{2}$$

where $y(x)$ is the true class of x, and $\widehat{y}$ is the predicted class.

Distributional Generalization: *Let f be a classifier trained on TrainSet. Then f satisfies Distributional Generalization with respect to tests $\mathcal{T}$ if:*

$$\forall T\in\mathcal{T}: \quad \mathop{\mathbb{E}}_{\substack{x\sim TrainSet\\ \widehat{y}\leftarrow f(x)}}[T(x,\widehat{y})] \approx \mathop{\mathbb{E}}_{\substack{x\sim TestSet\\ \widehat{y}\leftarrow f(x)}}[T(x,\widehat{y})] \tag{3}$$

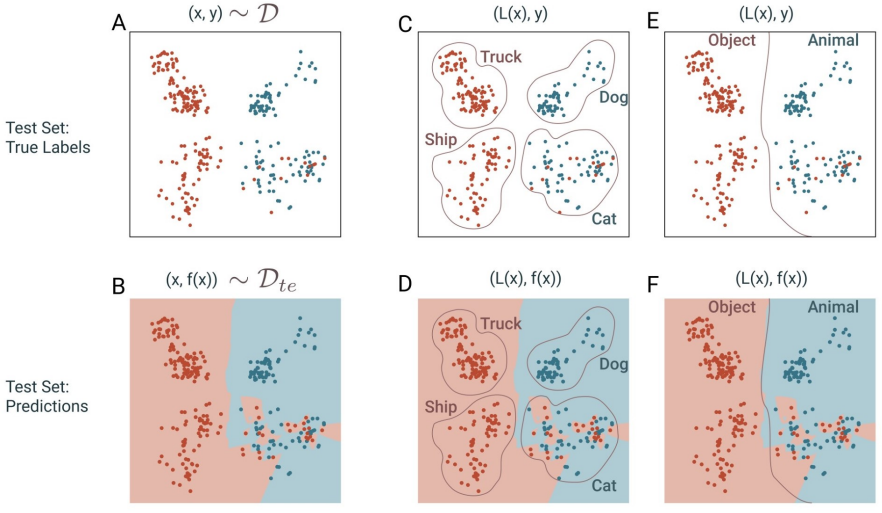
which we also write as

$$\mathcal{D}_{\text{tr}} \approx^{\mathcal{T}} \mathcal{D}_{\text{te}} \tag{4}$$

Conjecture 1 (Feature Calibration)**.** *For all natural distributions $\mathcal{D}$, number of samples n, family of interpolating models $\mathcal{F}$, and $\varepsilon \geq 0$, the following distributions are statistically close for all $(\varepsilon, \mathcal{F}, \mathcal{D}, n)$-distinguishable features L:*

$$\mathop{(L(x), f(x))}_{\substack{f\leftarrow \text{Train}_{\mathcal{F}}(\mathcal{D}^n)\\ x,y\sim\mathcal{D}}} \quad \approx_{\varepsilon} \quad \mathop{(L(x), y)}_{x,y\sim\mathcal{D}} \tag{7}$$

Feature Calibration



Constant Calibration

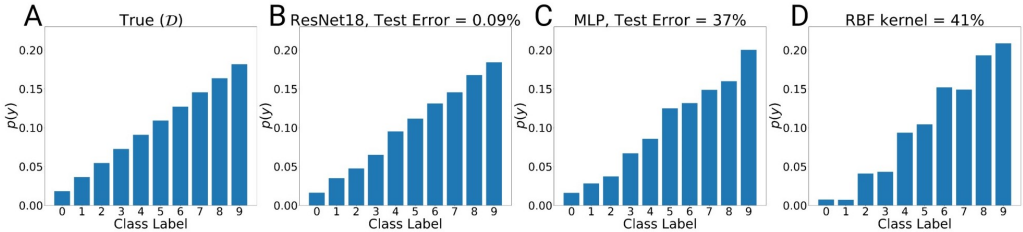


Figure 4: **Feature Calibration for Constant Partition L:** The CIFAR-10 train and test sets are class rebalanced according to (A). Interpolating classifiers are trained on the train set, and we plot the class balance of their outputs on the test set. This roughly matches the class balance of the train set, even for poorly-generalizing classifiers.

Neural network is not Bayes Optimal

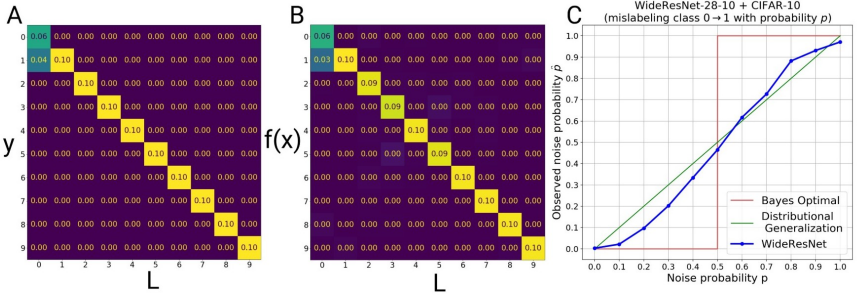


Figure 4: **Feature Calibration with original classes on CIFAR-10**: We train a WRN-28-10 on the CIFAR-10 dataset where we mislabel class $0 \to 1$ with probability p. (A): Joint density of the distinguishable features L (the original CIFAR-10 class) and the classification task labels y on the train set for noise probability $p = 0.4$. (B): Joint density of the original CIFAR-10 classes L and the network outputs $f(x)$ on the test set. (C): Observed noise probability in the network outputs on the test set (the (1, 0) entry of the matrix in B) for varying noise probabilities p

Detailed feature calibration between class (confusion matrix)

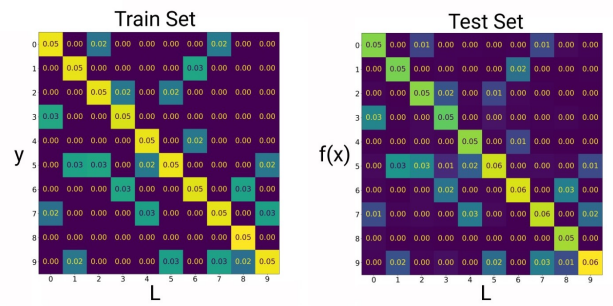


Figure 6: **Feature Calibration with random confusion matrix on CIFAR-10:** Left: Joint density of labels y and original class L on the train set. Right: Joint density of classifier predictions $f(x)$ and original class L on the test set, for a WideResNet28-10 trained to interpolation. These two joint densities are close, as predicted by Conjecture 1.

Detailed feature calibration inside class

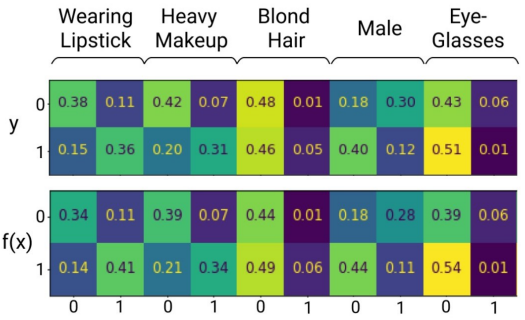
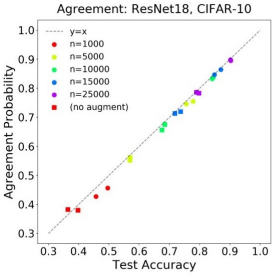


Figure 7: **Feature Calibration for multiple features on CelebA**: We train a ResNet-50 to perform binary classification task on the CelebA dataset. The top row shows the joint distribution of this task label with various other attributes in the dataset. The bottom row shows the same joint distribution for the ResNet-50 outputs on the test set. Note that the network was not given any explicit inputs about these attributes during training.

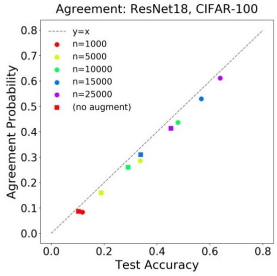
Model not very good? (Some true labels ⇔ noise): Agreement Property

Conjecture 2 (Agreement Property). *For certain classifier families $\mathcal{F}$ and distributions $\mathcal{D}$, the test accuracy of a classifier is close to its* agreement probability *with an independently-trained classifier. That is, let S_1, S_2 be independent train sets sampled from $\mathcal{D}^n$, and let f_1, f_2 be classifiers trained on S_1, S_2 respectively. Then*

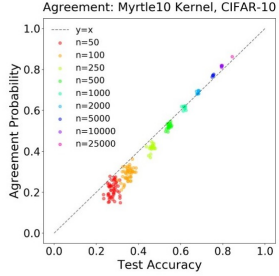
$$\Pr_{\substack{S_1 \sim \mathcal{D}^n \\ f_1 \leftarrow \mathrm{Train}_{\mathcal{F}}(S_1) \\ (x,y) \sim \mathcal{D}}} [f_1(x) = y] \approx \Pr_{\substack{S_1, S_2 \sim \mathcal{D}^n \\ f_i \leftarrow \mathrm{Train}_{\mathcal{F}}(S_i) \\ (x,y) \sim \mathcal{D}}} [f_1(x) = f_2(x)] \tag{12}$$



(a) ResNet18 on CIFAR-10.

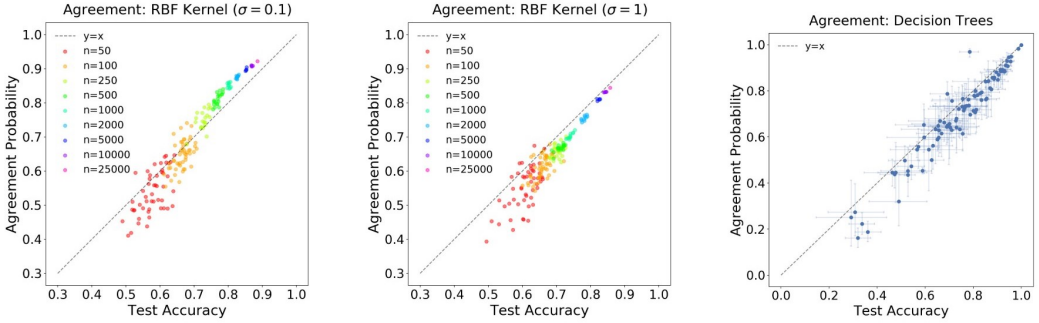


(b) ResNet18 on CIFAR-100.



(c) Myrtle Kernel on CIFAR-10.

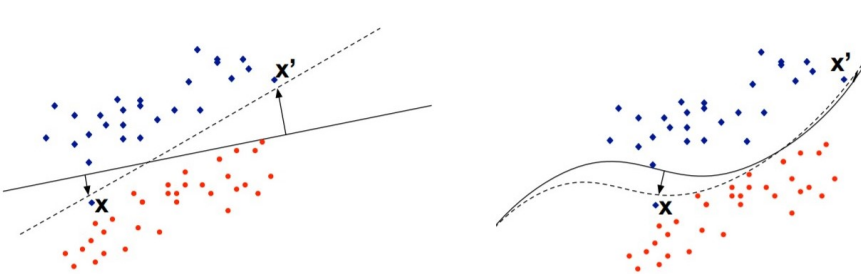
Agreement property also exists in other models



(a) RBF on Fashion-MNIST (b) RBF on Fashion-MNIST (c) Decision Trees on UCI

Figure 9: **Agreement Property for RBF and decision trees.** For two classifiers trained on disjoint train sets, the probability they agree with each other (on the test set) is close to their test accuracy. For UCI, each point corresponds to one UCI task, and error bars show 95% Clopper-Pearson confidence intervals in estimating population quantities.

Local Elasticity[5] (also known as stiffness[6])



(a) Linear classifier updated by SGD. (b) Neural networks updated by SGD.

Figure 2: An illustration of linear regression and neural networks updated via SGD. The prediction of $\boldsymbol{x}'$ changes a lot after an SGD update on $\boldsymbol{x}$ in the linear case, though $\boldsymbol{x}'$ is far away from $\boldsymbol{x}$. In contrast, the change in the prediction at $\boldsymbol{x}'$ is rather small in the neural networks case.

[5] ICLR2020 https://arxiv.org/pdf/1910.06943.pdf
[6] https://arxiv.org/pdf/1901.09491.pdf

Relative similarity and kernelized similarity

$$\boldsymbol{w}^+ = \boldsymbol{w} - \eta \frac{\mathrm{d}\mathcal{L}(f(\boldsymbol{x}, \boldsymbol{w}), y)}{\mathrm{d}\boldsymbol{w}} = \boldsymbol{w} - \eta \frac{\partial \mathcal{L}(f(\boldsymbol{x}, \boldsymbol{w}), y)}{\partial f} \cdot \frac{\partial f(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}}. \tag{1}$$

The essence of local elasticity is that the change in the prediction has an (approximate) monotonic relationship with the similarity of feature vectors. Therefore, the change can serve as a *proxy* for the similarity of two inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$:

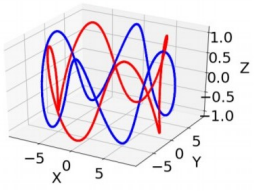
$$S_{\mathrm{rel}}(\boldsymbol{x}, \boldsymbol{x}') := \frac{|f(\boldsymbol{x}', \boldsymbol{w}^+) - f(\boldsymbol{x}', \boldsymbol{w})|}{|f(\boldsymbol{x}, \boldsymbol{w}^+) - f(\boldsymbol{x}, \boldsymbol{w})|}. \tag{2}$$

$$\begin{aligned} f(\boldsymbol{x}', \boldsymbol{w}^+) - f(\boldsymbol{x}', \boldsymbol{w}) &= f\left(\boldsymbol{x}', \boldsymbol{w} - \eta \frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}}\right) - f(\boldsymbol{x}', \boldsymbol{w}) \\ &\approx f(\boldsymbol{x}', \boldsymbol{w}) - \left\langle \frac{\partial f(\boldsymbol{x}', \boldsymbol{w})}{\partial \boldsymbol{w}}, \eta \frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}} \right\rangle - f(\boldsymbol{x}', \boldsymbol{w}) \\ &= -\eta \frac{\partial \mathcal{L}}{\partial f} \left\langle \frac{\partial f(\boldsymbol{x}', \boldsymbol{w})}{\partial \boldsymbol{w}}, \frac{\partial f(\boldsymbol{x}, \boldsymbol{w})}{\partial \boldsymbol{w}} \right\rangle. \end{aligned}$$

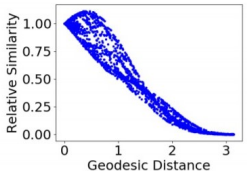
The factor $-\eta \frac{\partial \mathcal{L}}{\partial f}$ does not involve $\boldsymbol{x}'$, just as the denominator $|f(\boldsymbol{x}, \boldsymbol{w}^+) - f(\boldsymbol{x}, \boldsymbol{w})|$ in Equation (2). This observation motivates an alternative definition of the similarity:

$$S_{\mathrm{ker}}(\boldsymbol{x}, \boldsymbol{x}') := \frac{f(\boldsymbol{x}', \boldsymbol{w}) - f(\boldsymbol{x}', \boldsymbol{w}^+)}{\eta \frac{\partial \mathcal{L}(f(\boldsymbol{x}, \boldsymbol{w}), y)}{\partial f}}. \tag{3}$$

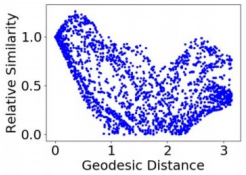
Comparisons between ReLU neural networks and linear neural networks



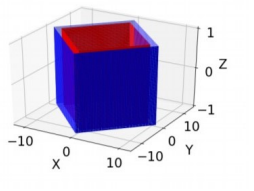
(a) The torus.



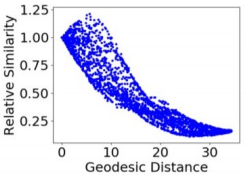
(b) Two-layer nets fitting the torus.



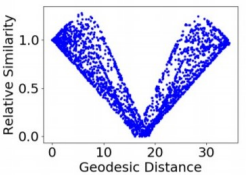
(c) Two-layer *linear* nets fitting the torus.



(d) The two folded boxes.



(e) Three-layer nets fitting the boxes.



(f) Three-layer *linear* nets fitting the boxes.

Elastic-Locality based clustering

Algorithm 1 The Local Elasticity Based Clustering Algorithm.

Input: primary dataset $\mathcal{P} = \{\boldsymbol{x}_i\}_{i=1}^n$, auxiliary dataset $\mathcal{A} = \{\widetilde{\boldsymbol{x}}_j\}_{j=1}^m$, classifier $f(\boldsymbol{x}, \boldsymbol{w})$, initial weights $\boldsymbol{w}_0$, loss function $\mathcal{L}$, learning rate η_t, option $o \in \{\texttt{relative}, \texttt{kernelized}\}$

combine $\mathcal{D} = \{(\boldsymbol{x}_i, y_i = 1) \text{ for } \boldsymbol{x}_i \in \mathcal{P}\} \bigcup \{(\boldsymbol{x}_i, y_i = -1) \text{ for } \boldsymbol{x}_i \in \mathcal{A}\}$
set S to $n \times n$ matrix of all zeros
for $t = 1$ to $n + m$ **do**
    sample $(\boldsymbol{x}, y)$ from $\mathcal{D}$ w/o replacement
    $\boldsymbol{w_t} = \texttt{SGD}(\boldsymbol{w}_{t-1}, \boldsymbol{x}, y, f, \mathcal{L}, \eta_t)$
    if $y = 1$ **then**
        $\boldsymbol{p}_t = \texttt{Predict}(\boldsymbol{w}_t, \mathcal{P}, f)$
        find $1 \le i \le n$ such that $\boldsymbol{x} = \boldsymbol{x}_i \in \mathcal{P}$
        if $o = \texttt{relative}$ **then**
            $\boldsymbol{s}_t = \frac{|\boldsymbol{p}_t - \boldsymbol{p}_{t-1}|}{|\boldsymbol{p}_t(i) - \boldsymbol{p}_{t-1}(i)|}$
        else
            $g_t = \texttt{GetGradient}(\boldsymbol{w}_{t-1}, \boldsymbol{x}, y, f, \mathcal{L})$
            $\boldsymbol{s}_t = \frac{\boldsymbol{p}_t - \boldsymbol{p}_{t-1}}{-\eta_t \times g_t}$
        end if
    end if
    set the ith row $S(i, :) = \boldsymbol{s}_t$
end for
$S_{\text{symm}} = \frac{1}{2}(S + S^\top)$
$\boldsymbol{y}_{\text{subclass}} = \text{Clustering}(S_{\text{symm}})$
return $\boldsymbol{y}_{\text{subclass}}$

Locally Elastic Stability[7]

Definition (Locally Elastic Stability)

An algorithm $\mathcal{A}$ has locally elastic stability $\beta_m(\cdot,\cdot)$ with respect to the loss function l if, for all m, the inequality

$$|l(\mathcal{A}_S, z) - l(\mathcal{A}_{S^{\setminus i}}, z)| \leq \beta_m(z_i, z)$$

holds for all $S \in \mathcal{Z}^m, 1 \leq i \leq m$, and $z \in \mathcal{Z}$.

Definition (Uniform Stability)

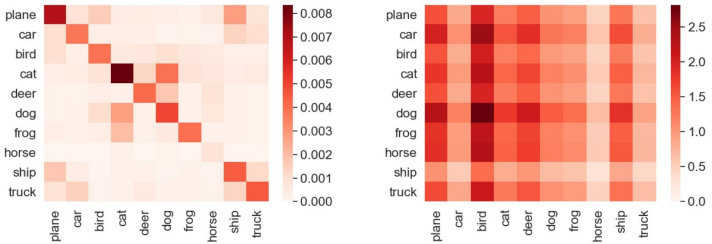
Let β_m^{U} be a sequence of scalars. An algorithm $\mathcal{A}$ has uniform stability β_m^{U} with respect to the loss function l if

$$|l(\mathcal{A}_S, z) - l(\mathcal{A}_{S^{\setminus i}}, z)| \leq \beta_m^{\mathsf{U}} \tag{1}$$

holds for all $S \in \mathcal{Z}^m, 1 \leq i \leq m$, and $z \in \mathcal{Z}$.

[7] AISTATS 2021? https://arxiv.org/pdf/2010.13988.pdf

Comparisons between ReLU neural networks and linear neural networks

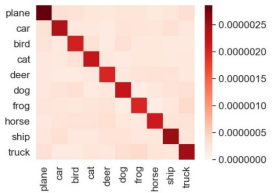


(a) Sensitivity of neural networks.

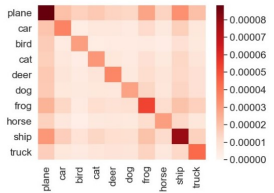
(b) Sensitivity of a linear model.

Figure 1: Class-level sensitivity approximated by influence functions for an 18-layer ResNet and a linear model on CIFAR-10. The vertical axis denotes the classes in the test data and the horizontal axis denotes the classes in the training data. The class-level sensitivity from class a in the training data to class b in the test data is defined as $C(c_a, c_b) = \frac{1}{|S_a| \times |\tilde{S}_b|} \sum_{z_i \in S_a} \sum_{z \in \tilde{S}_b} |l(\hat{\theta}, z) - l(\hat{\theta}^{-i}, z)|$, where S_a denotes the set of examples from class a in the training data and $\tilde{S}_b$ denotes set of examples from class b in the test data. More details are in Appendix B.

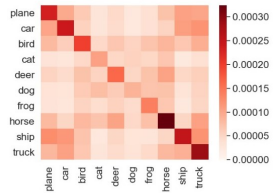
Comparisons between ReLU neural networks and linear neural networks



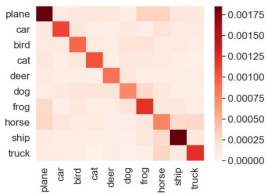
(a) Neural networks (epoch 0).



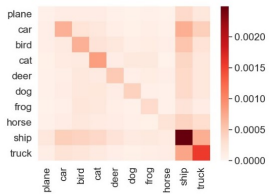
(b) Neural networks (epoch 10).



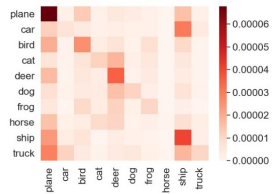
(c) Neural networks (epoch 50).



(d) Linear model (epoch 0).



(e) Linear model (epoch 10).



(f) Linear model (epoch 50).

Generalization Bounds

We write $\Delta(\mathcal{A}_S)$ as a shorthand for the defect $\mathbb{E}_z l(\mathcal{A}_S, z) - \sum_{j=1}^m l(\mathcal{A}_S, z_j)/m$, where the expectation $\mathbb{E}_z$ is over the randomness embodied in $z \sim \mathcal{D}$. In particular, $\mathbb{E}_z l(\mathcal{A}_S, z)$ depends on $\mathcal{A}_S$.

Theorem

Let $\mathcal{A}$ be an algorithm that has locally elastic stability $\beta_m(\cdot,\cdot)$ with respect to the loss function l. Fixing $0 < \delta < 1$ and $\eta > 0$, for sufficiently large m, with probability at least $1-\delta$, we have

$$\Delta(\mathcal{A}_S) \leq \frac{2\sup_{z' \in \mathcal{Z}} \mathbb{E}_z \beta(z', z)}{m} + 2\left(2\sup_{z' \in \mathcal{Z}} \mathbb{E}_z \beta(z', z) + \eta + M_l\right)\sqrt{\frac{2\log(2/\delta)}{m}}.$$

Models	$\sup_{z' \in S, z \in \mathcal{Z}} \beta_m(z', z)$	$\sup_{z' \in \mathcal{Z}} \mathbb{E}_z \beta_m(z', z)$	*ratio*
Linear model	314	40	8
Neural networks	3.05	0.02	153

Generalization Bounds for SGD

Definition

A randomized algorithm $\mathcal{A}$ is $\beta_m(\cdot,\cdot)$-locally elastic stable if for all $S \in \mathcal{Z}^n$, we have

$$|\mathbb{E}_{\mathcal{A}}[l(\mathcal{A}_S, z)] - \mathbb{E}_{\mathcal{A}}[l(\mathcal{A}_{S^{\setminus i}}, z)]| \leq \beta_m(z_i, z),$$

where the expectation is over the randomness embedded in the algorithm $\mathcal{A}$.

Proposition (Non-convex Optimization)

Assume that the loss function $l(\cdot, z)$ is non-negative and bounded for all $z \in \mathcal{Z}$. Without loss of generality, we assume $0 \leq l(\cdot, z) \leq 1$. In addition, we assume $l(\cdot, z)$ is α-smooth and convex for all $z \in \mathcal{Z}$. We further assume $l(\cdot, z)$ is $L(z)$-Lipschitz and $L(z) < \infty$ for all $z \in \mathcal{Z}$ and $L = \sup_{z \in \mathcal{Z}} L(z) < \infty$. Suppose that we run SGD for T steps with monotonically non-increasing learning rate $\eta_t \leq c/t$ for some constant $c > 0$. Then,

$$|\mathbb{E}[l(\widehat{\theta}_T, z)] - \mathbb{E}[l(\widehat{\theta}^{\setminus i}_T, z)]| \leq \gamma_m \Big(c(L(z_i) + L)L(z)\, T^{\alpha c} \Big)^{\frac{1}{\alpha c + 1}},$$

where $\gamma_m = (1 + 1/(\alpha c))/(m-1)$.