Understanding Deep Learning Requires Rethinkinking Generalization

Paulina Castillo, Ricardo Marino & Sebastian Molina

Introduction

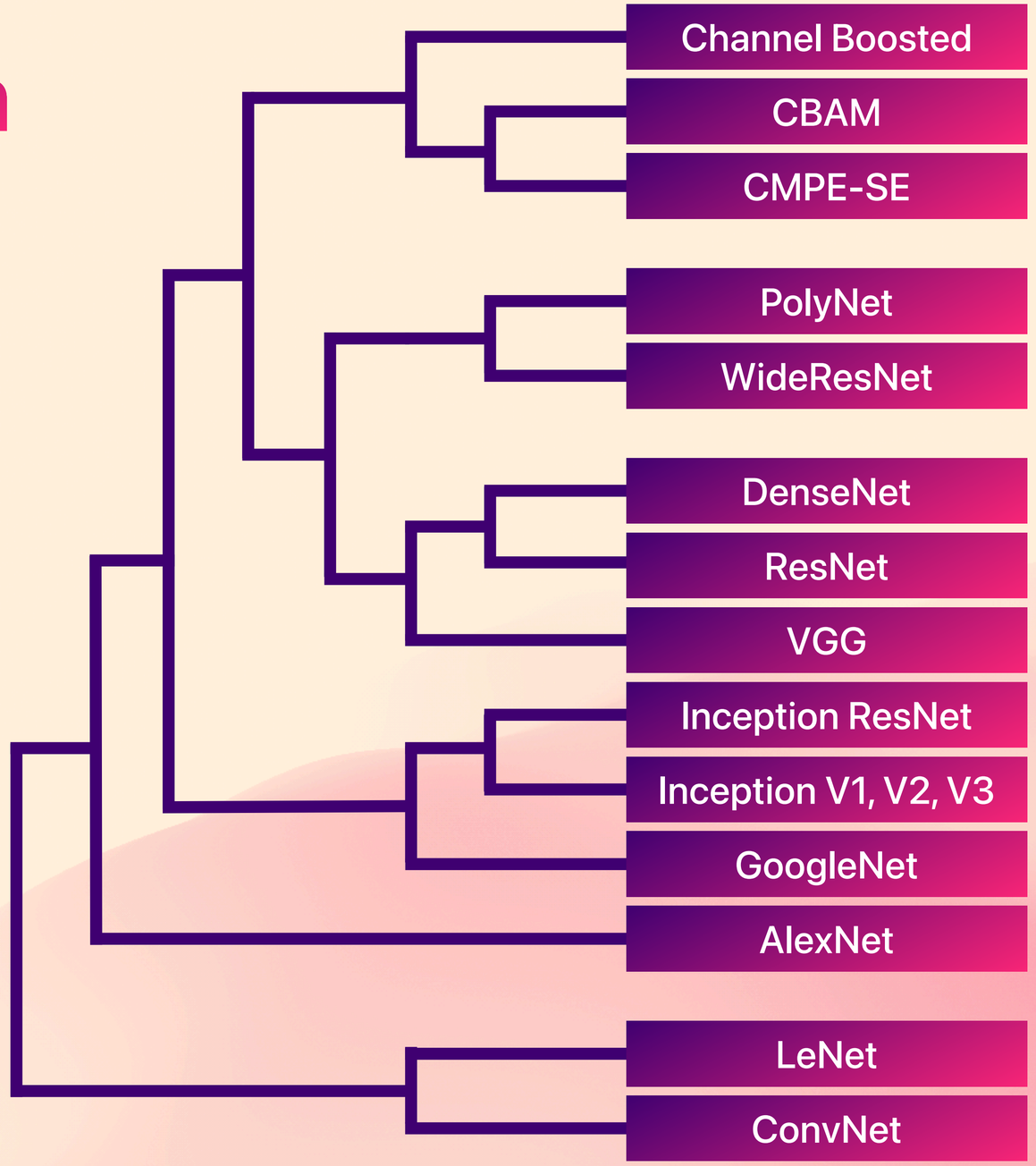


Table 2 Performance comparison of the recent architectures of different categories. Top 5 error rate is reported for all architectures

Architecture name	Year	Main contribution	Parameters	Error rate	Depth	Category	References
LeNet	1998	First popular CNN architecture	0.060 M	[dist]MNIST: 0.8 MNIST: 0.95	5	Spatial exploitation	LeCun et al. (1995)
AlexNet	2012	Deeper and wider than the LeNet Uses Relu, dropout and overlap Pooling GPUs NVIDIA GTX 580	60 M	ImageNet: 16.4	8	Spatial Exploitation	Krizhevsky et al. (2012)
ZfNet	2014	Visualization of intermediate layers	60 M	ImageNet: 11.7	8	Spatial exploitation	Zeiler and Fergus (2013)
VGG	2014	Homogenous topology Uses small size kernels	138 M	ImageNet: 7.3	19	Spatial exploitation	Simonyan and Zisserman (2015)
GoogLeNet	2015	Introduced block concept Split transform and merge idea	4 M	ImageNet: 6.7	22	Spatial exploitation	Szegedy et al. (2015)
Inception-V3	2015	Handles the problem of a representational bottleneck Replace large size filters with small filters	23.6 M	ImageNet: 3.5 Multi-crop: 3.58 Single-Crop: 5.6	159	Depth + width	Szegedy et al. (2016b)
Highway networks	2015	Introduced an idea of multi-path	2.3 M	CIFAR-10: 7.76	19	Depth + multi-path	Srivastava et al. (2015a)
Inception-V4	2016	Split transform and merge idea Uses asymmetric filters	35 M	ImageNet: 4.01	70	Depth +width	Szegedy et al. (2016a)
Inception-ResNet	2016	Uses split transform merge idea and residual links	55.8 M	ImageNet: 3.52	572	Depth + width + multi-path	Szegedy et al. (2016a)
ResNet	2016	Residual learning Identity mapping based skip connections	25.6 M 1.7 M	ImageNet: 3.6 CIFAR-10: 6.43	152 110	Depth + multi-path	He et al. (2015a)
DelugeNet	2016	Allows cross layer information flow in deep networks	20.2 M	CIFAR-10: 3.76 CIFAR-100: 19.02	146	Multi-path	Kuen et al. (2018)

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review, 53(8), 5455–5516. doi:10.1007/s10462-020-09825-6

But...

What distinguishes Neural Networks that generalize well from those that don't?

Inception V3

Random Deep Neural Network

Statistical Learning Theory

Rademacher
Complexity

VC Dimension

Uniform
Stability

Statistical Learning Theory

VC Dimension

Rademacher Complexity

Uniform Stability

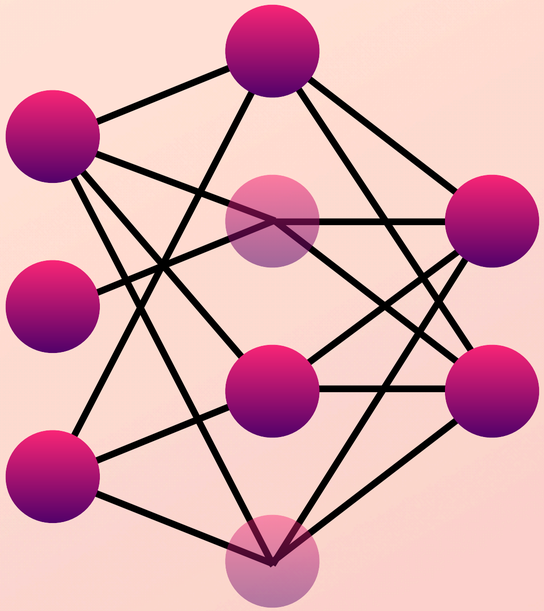
Statistical Learning Theory

Rademacher Complexity

Uniform Stability

VC Dimension

Regularization



Dropout

Data augmentation

$$\min \left[\text{cost function} + \frac{\lambda}{2} \| w \|_{l_2}^2 \right]$$

Weight decay

Methodology: Label Replacement

Dog

Methodology: Label Replacement

Dog

With probability
1-p still a dog

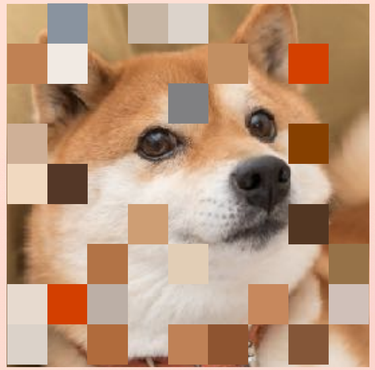
Methodology: Label Replacement

Dog

With probability
1-p still a dog

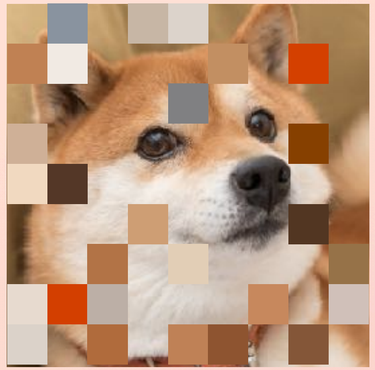
Cat

Methodology: Pixel Replacement

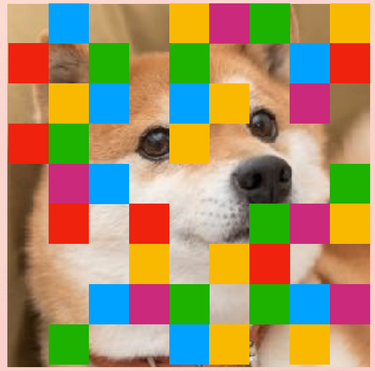


Pixel
Shuffle

Methodology: Pixel Replacement

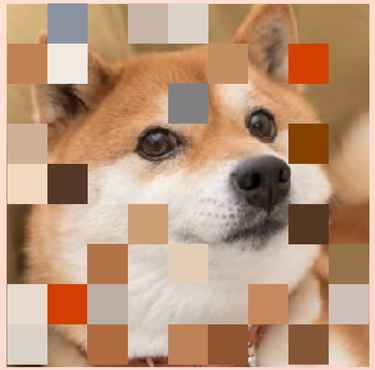


Pixel
Shuffle

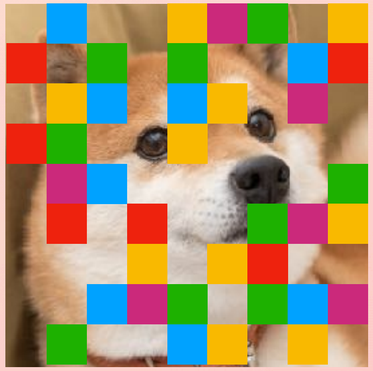


Random
Pixels

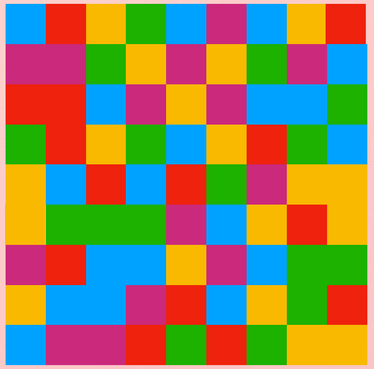
Methodology: Pixel Replacement



Pixel Shuffle

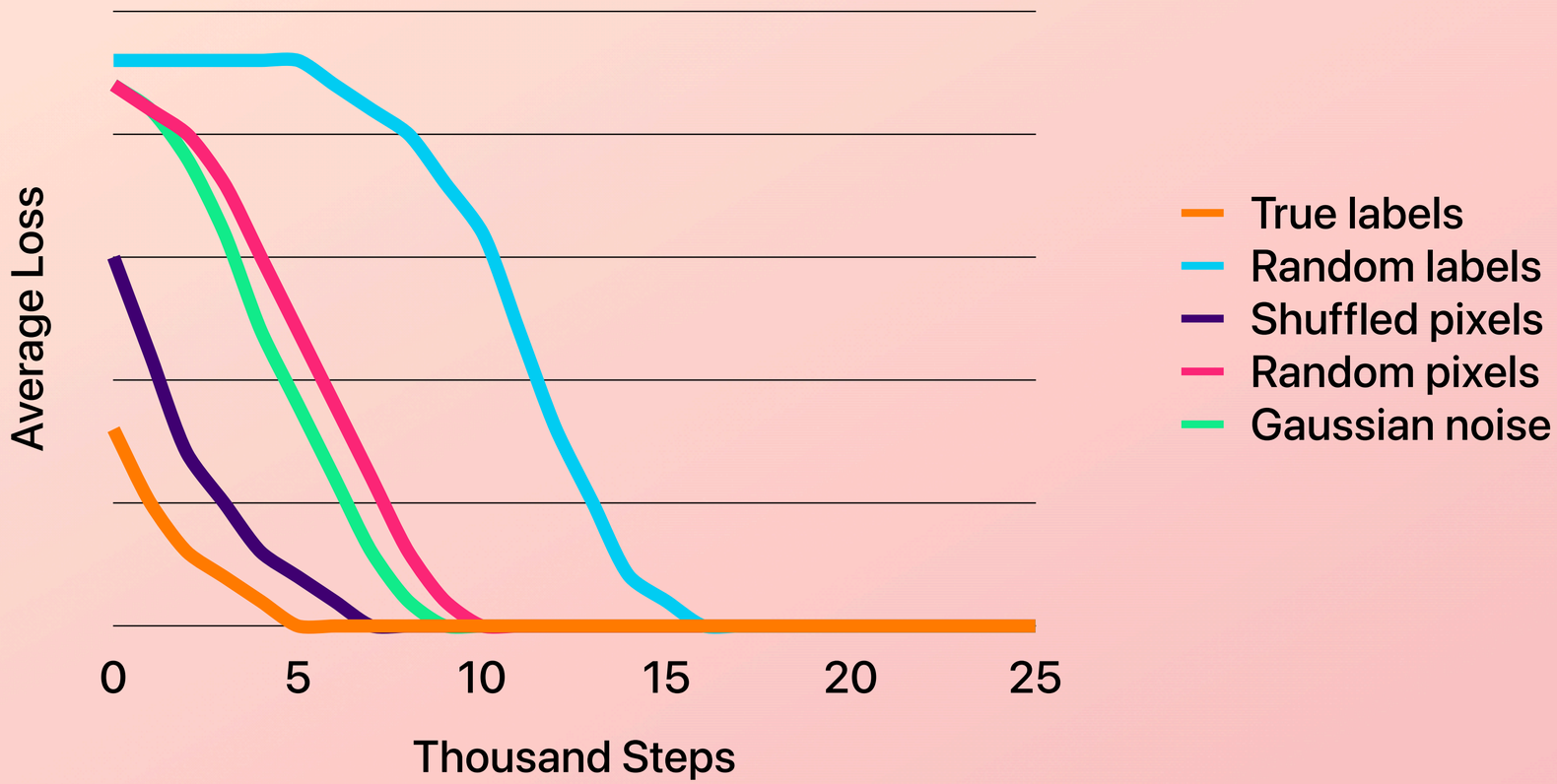


Random Pixels

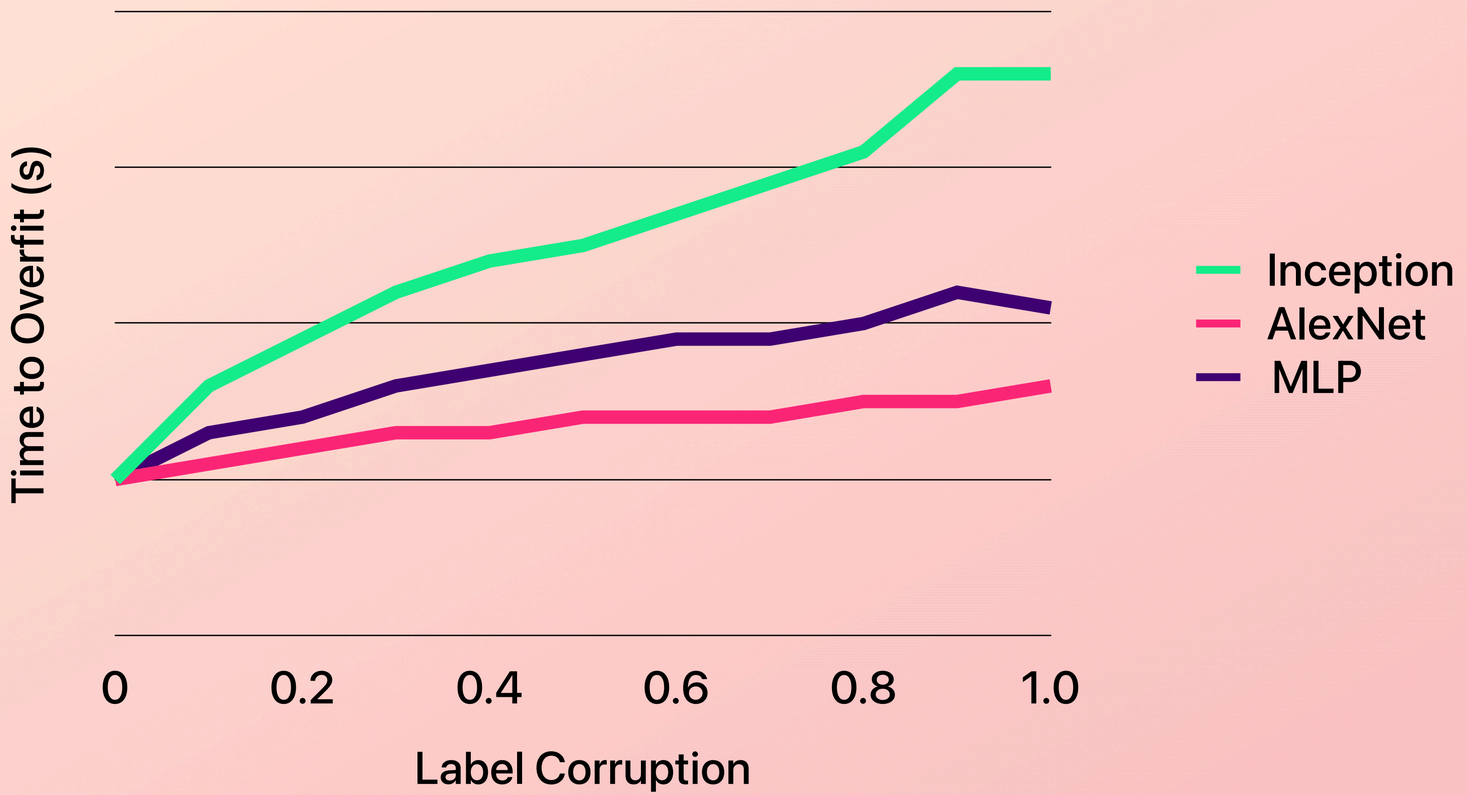


Gaussian Noise

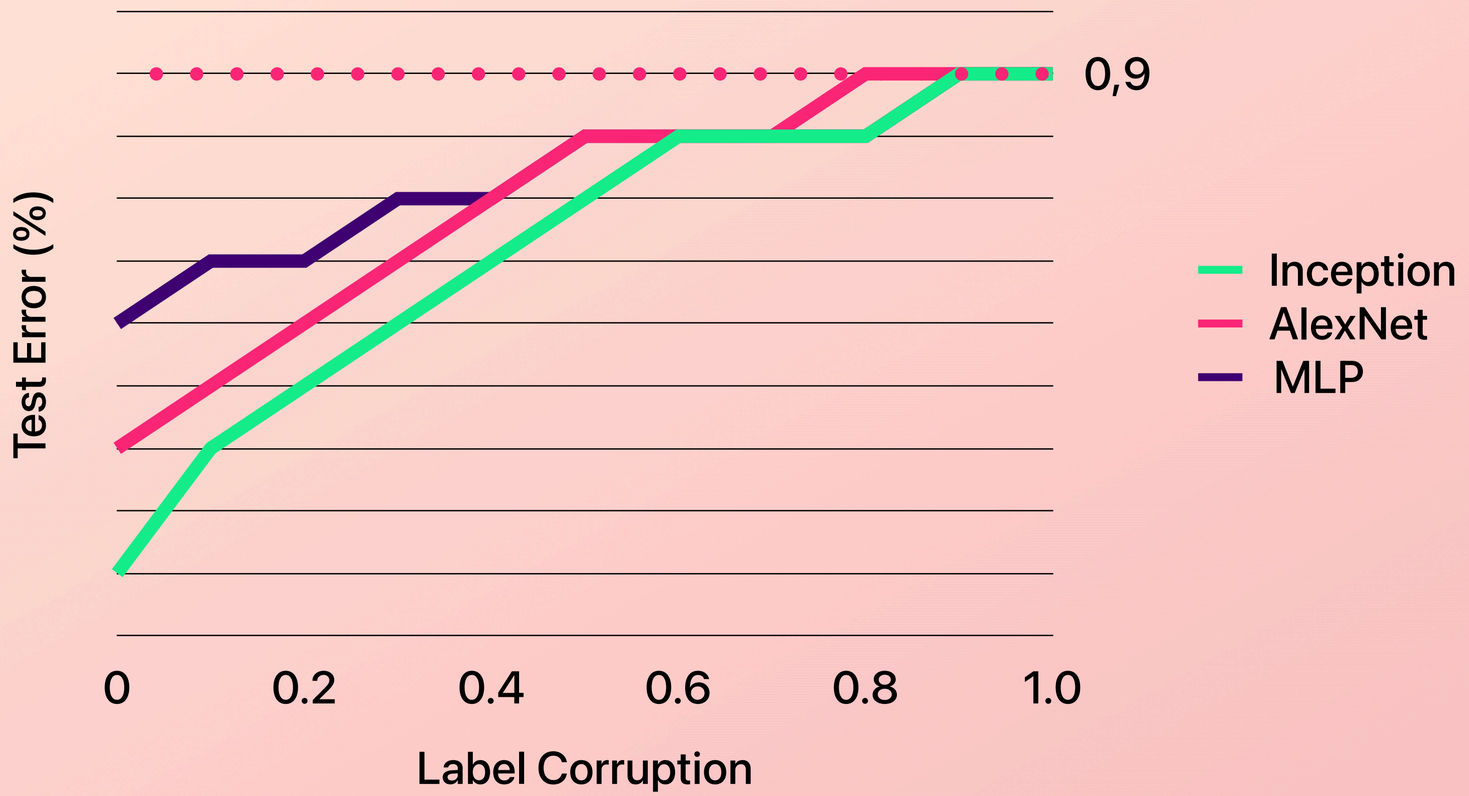
Results


Average Loss
0
5
10
15
20
25
Thousand Steps
True labels
Random labels
Shuffled pixels
Random pixels
Gaussian noise

Results


Time to Overfit (s)
0
0.2
0.4
0.6
0.8
1.0
Label Corruption
Inception
AlexNet
MLP

Results


0,9
Test Error (%)
0
0.2
0.4
0.6
0.8
1.0
Label Corruption
Inception
AlexNet
MLP

Rademacher Complexity

$$E_{\text{in}}(h) = \frac{1}{N} \sum_{i=1}^{N} [h(x_i) \neq f(x_i)]$$

Rademacher Complexity

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{i=1}^{N}[h(x_i) \neq f(x_i)]$$

$$= \frac{1}{N}\sum_{i=1}^{N}\frac{1 - f(x_i)h(x_i)}{2}$$

Rademacher Complexity

$$
\begin{aligned}
E_{\text{in}}(h) &= \frac{1}{N}\sum_{i=1}^{N}[h(x_i) \neq f(x_i)] \\
&= \frac{1}{N}\sum_{i=1}^{N}\frac{1 - f(x_i)h(x_i)}{2} \\
&= \frac{1}{2} - \frac{1}{2N}\sum_{i=1}^{N} f(x_i)h(x_i)
\end{aligned}
$$

Rademacher Complexity

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{i=1}^{N}[h(x_i) \neq f(x_i)]$$

$$= \frac{1}{N}\sum_{i=1}^{N}\frac{1 - f(x_i)h(x_i)}{2}$$

$$= \frac{1}{2} - \frac{1}{2N}\sum_{i=1}^{N} f(x_i)h(x_i)$$

Correlation

$$\arg\max_h \frac{1}{N} \sum_{i=1}^{N} f(x_i) h(x_i)$$

Minimizing training error is maximizing
correlation between labels and hypotheses.

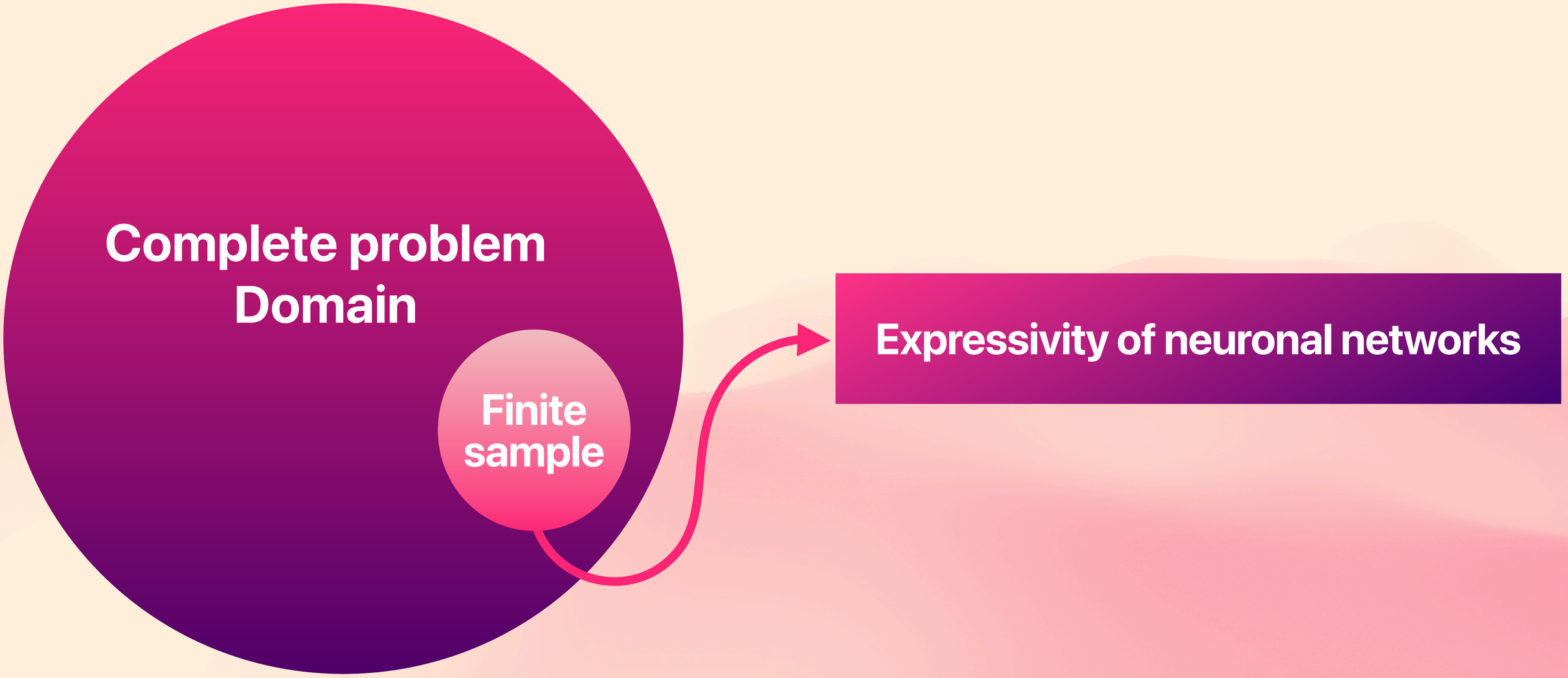
$$\widehat{\mathfrak{R}}_n(\mathcal{H}) = \mathbb{E}_\sigma \left[\sup_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \sigma_i h(x_i) \right]$$

Swapping labels by Rademacher's random variables to obtain Rademacher's Complexity.

Implications

- **Training** in many neural networks **suggest** that they can **adjust perfectly** to **training data** with **random labels**.
- In this case, **Rademacher complexity** is approximately **1**, which is a **trivial upper bound** for this complexity.
- This complexity **measure** is **not able** to derive **useful** results on **generalization**.
- By a similar reason, **VC dimension** is **not useful** in some cases.

Finite-Sample Expressivity



Finite-Sample Expressivity

Expressivity of neuronal networks

There exists a two-layer neural network with ReLU activations and
2n+d weights that can represent any function
on a sample of size ***n*** in ***d*** dimensions.

Implicit Regulation: An appeal to linear models

Samples

$$\{(x_1, y_2), (x_2, y_2), \ldots, (x_n, y_n)\}$$

Loss function

$$\text{loss} : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}^d; \quad \text{loss}(y, y) = 0$$

Empirical Risk Minimization problem

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \text{loss}\left(w^T x_i, y_i\right)$$

Empirical Risk Minimization problem

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & & \vdots \\ x_1^n & x_2^n & \cdots & x_d^n \end{bmatrix}$$

Empirical Risk Minimization problem

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & & \vdots \\ x_1^n & x_2^n & \cdots & x_d^n \end{bmatrix}$$

$$\text{rank}(X) = n$$

$$Xw = y$$

Infinite solutions!

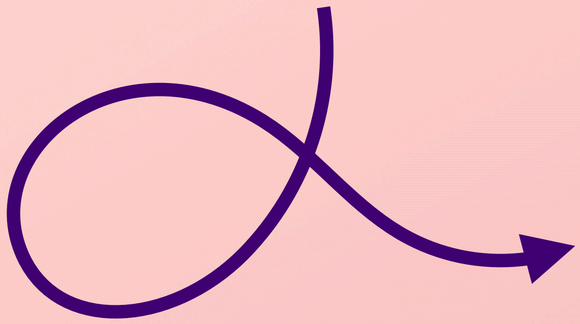
Empirical Risk Minimization problem

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & & \vdots \\ x_1^n & x_2^n & \cdots & x_d^n \end{bmatrix}$$

$$\text{rank}(X) = n$$

$$Xw = y$$

Infinite solutions!



$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \text{loss}\left(w^T x_i, y_i\right)$$

Is there a way to determine when one global minimum will generalize whereas another will not?

Stochastic Gradient Descent and *Kernel trick*

Update rule: $w_{t+1} = w_t - \eta_t e_t x_{i_t}$

$w_0 = 0$

Stochastic Gradient Descent and *Kernel trick*

Update rule: $w_{t+1} = w_t - \eta_t e_t x_{i_t}$

$w_0 = 0$ —**SGD**→ $w = \sum_{i=1}^{n} \alpha_i x_i$

$w = X^{\mathrm{T}} \alpha$

Stochastic Gradient Descent and *Kernel trick*

Update rule: $w_{t+1} = w_t - \eta_t e_t x_{i_t}$

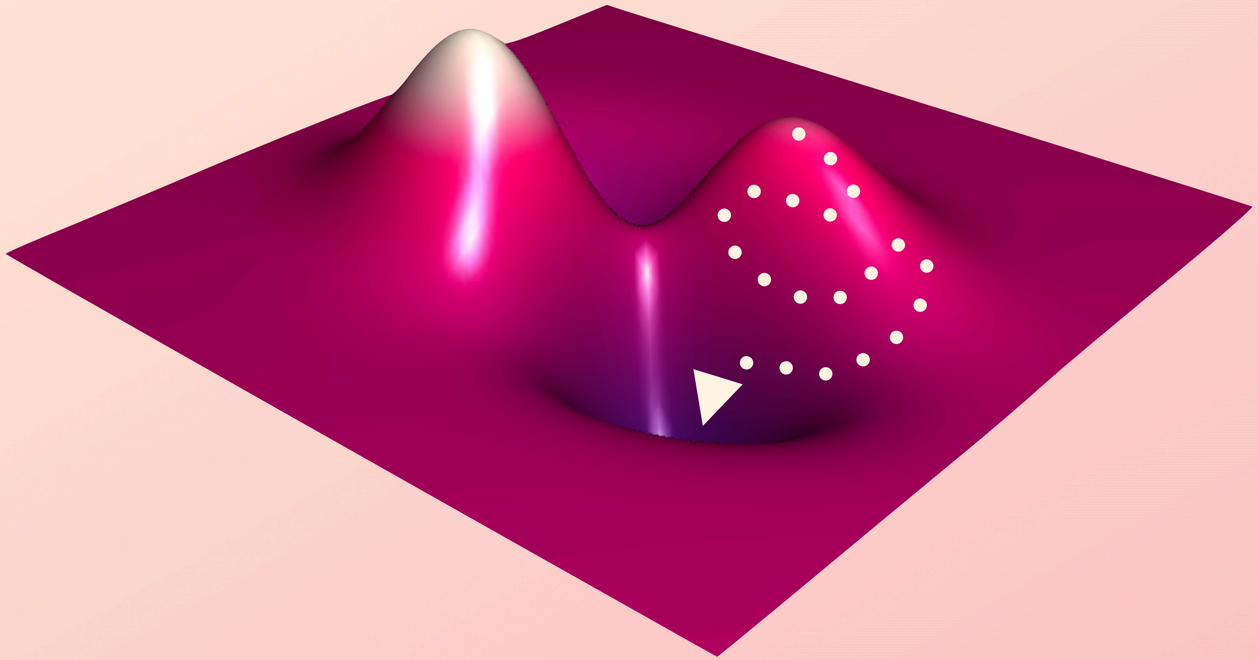
$$w_0 = 0 \xrightarrow{\text{SGD}} w = \sum_{i=1}^{n} \alpha_i x_i \xrightarrow{\text{*Interpolate } y \text{ so that*}} Xw = y$$

$$w = X^{\mathrm{T}}\alpha$$

$$XX^{\mathrm{T}}\alpha = y$$

Stochastic Gradient Descent and *Kernel trick*

Unique solution: $XX^{\mathrm{T}}\alpha = y$



Some results

data set	pre-processing	test error
MNIST	none	1.2%
MNIST	gabor filters	0.6%
CIFAR10	none	46%
CIFAR10	random conv-net	17%

Generalizing with kernel trick. The test error associated with solving the kernel equation on small benchmarks.

Note that changing the preprocessing can significantly change the resulting test error.

Kernel trick

Minimum l_2 norm

$$Xw = y$$

data set	pre-processing	test error
MNIST	none	1.2%
MNIST	gabor filters	0.6%
CIFAR10	none	46%
CIFAR10	random conv-net	17%

Conclusions

- The **effective capacity** of several neuronal networks is **large** enough to **shatter** the **training data**.

- Statistical Learning Theory and **traditional measures struggle** to **explain** the **generalization** ability of even simple linear models.