So why deep learning works?

Taeyoung Kim

Korea Advanced Institute of Science and Technology

mekty2012@kaist.ac.kr

May 1, 2021

Overview

1. Why does neural net work?
   - Universal Approximation Theorem
   - Deep Visualization
   - Manifold Hypothesis
   - Finite Approximation
2. Why does gradient descent work?
   - Global Minimum
   - Generalizability
3. How can I trust Deep Learning?
   - Verifying Neural Network
   - Testing Neural Network

Universal Approximation Theorem-1

Neural Network is dense in function space, so we can approximate any function with neural network.
What Neural Network, What function space, What topology?

Universal Approximation Theorem-2

We say a function $\sigma : \mathbb{R} \to \mathbb{R}$ is sigmoidal if

$$\sigma(t) \to \begin{cases} 1 & t \to +\infty \\ 0 & t \to -\infty \end{cases}$$

For $y_i \in \mathbb{R}^n$ and $\alpha_i, \theta_i \in \mathbb{R}$ $(i = 1, \ldots, N)$, a neural network is a function from $\mathbb{R}^n$ to $\mathbb{R}$

$$G(x) = \sum_{i=1}^{N} \alpha_i \sigma(y_i^T x + \theta_i)$$

Universal Approximation Theorem-3

Theorem

Let σ be any continuous sigmoidal function. Then the neural networks

$$G(x) = \sum_{i=1}^{N} \alpha_i \sigma(y_i^T x + \theta_i)$$

are dense in $C(I_n)$.
In other words, given any $f \in C(I_n)$ and $\epsilon > 0$, there is a nerual network $G(x)$ such that

$$|G(x) - f(x)| < \epsilon$$

for every $x \in I_n$. [Cyb89]

Universal Approximation Theorem-4

We have other results also. Followings hold.

Function Types	Activation	Function space
$\sigma(y^T x + \theta)$	sigmoidal	$C(I_n)$
$\sigma(y^T x + \theta)$	$L^1(\mathbb{R})$, $\int \sigma(t)dt \neq 0$	$L^1(I_n)$
$\sigma(y^T x + \theta)$	continuous, sigmoidal	$L^2(I_n)$
$\sigma(Ux + y)$	Indicator of a rectangle	$L^1(I_n)$
$\sigma(tx + y)$	$L^1(\mathbb{R}^n)$, $\int \sigma(t)dt \neq 0$	$L^1(\mathbb{R}^n)$

Figure: Table of UAT variations

Meaning, given activation function, then finite linear sum of function types are dense in function spaces.

Universal Approximation Theorem-5

Theorem

For any Bochner-Lebesgue p-integrable function $f : \mathbb{R}^n \to \mathbb{R}^m$ *and* $\epsilon > 0$, *there exists a fully-connected ReLU network* F *of width exactly* $\max n + 1, m$ *with arbitrary depth that*

$$\int_{\mathbb{R}^n} ||f(x) - F(x)||^p dx < \epsilon$$

Moreover, there exists a function $f : \mathbb{R}^n \to \mathbb{R}^m$ *that is not approximable by fully-connected ReLU network* F *of width less than* $\max n + 1, m$. *[Par+21]*

Universal Approximation Theorem-6

But we usually have finite samples from function, not function itself. Here is finite sample version of universal approximation theorem.

Theorem

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. [Zha+16]

Universal Approximation Theorem-7

What about other networks?

CNN Continuous function on compact subset of $\mathbb{R}^n$ can be approximated by CNN w.r.t. supnorm. [Zho20]

RNN Dynamical system with continuous transition function can be approximated by RNN. [SZ06]

Generative Network For source and target probability measure π, p_z there is a neural network whose feed forward measure of π approximates p_z. (Constructive) [LL20]

References I

G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1989), pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274.

Yulong Lu and Jianfeng Lu. *A Universal Approximation Theorem of Deep Neural Networks for Expressing Probability Distributions*. 2020. arXiv: 2004.08867 [cs.LG].

Sejun Park et al. "Minimum Width for Universal Approximation". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=O-XJwyoIF-k.

Anton Maximilian Schäfer and Hans Georg Zimmermann. "Recurrent Neural Networks Are Universal Approximators". In: *Proceedings of the 16th International Conference on Artificial Neural Networks - Volume Part I*. ICANN'06. Athens, Greece: Springer-Verlag, 2006, pp. 632–640. ISBN: 3540386254. DOI: 10.1007/11840817_66. URL: https://doi.org/10.1007/11840817_66.

References II

Chiyuan Zhang et al. "Understanding deep learning requires rethinking generalization". In: *CoRR* abs/1611.03530 (2016). arXiv: 1611.03530. URL: http://arxiv.org/abs/1611.03530.

Ding-Xuan Zhou. "Universality of deep convolutional neural networks". In: *Applied and Computational Harmonic Analysis* 48.2 (2020), pp. 787–794. ISSN: 1063-5203. DOI: https://doi.org/10.1016/j.acha.2019.06.004. URL: https://www.sciencedirect.com/science/article/pii/S1063520318302045.

Deep Visualization-1

- Is there more intuitive way of understanding neural network?
- Can we visualize what neural network learned?
- What do each neurons do?

Deep Visualization-2

Recall the definition of neural network, where we have

$$x_{i,j} = \sigma\left(\sum_{k=1}^{N_{i-1}} w_{k,j} x_{i-1,k} + b_{i,j}\right)$$

Here each $x_{i,j}$ is called **activation value** $\alpha_{i,j}(x)$ of j-th node in layer i for input x.
If we find some input that $x_{i,j}$ is large enough, we can check what feature this node represents.

Deep Visualization-3

To increase $x_{i,j}$, we use gradient ascent, but regularized.
Initial input is defined as average of all input.

$$x_0 = \frac{1}{|D|} \sum_{x \in D} x$$

Then it is updated by gradient ascent, and some regularization.

$$x_{t+1} = \textit{Regularize}\left(x_t + \eta \frac{\partial \alpha_{i,j}(x)}{\partial x} \right)$$

After enough repeat, we have visualization of node. [Yos+15]

Deep Visualization-4

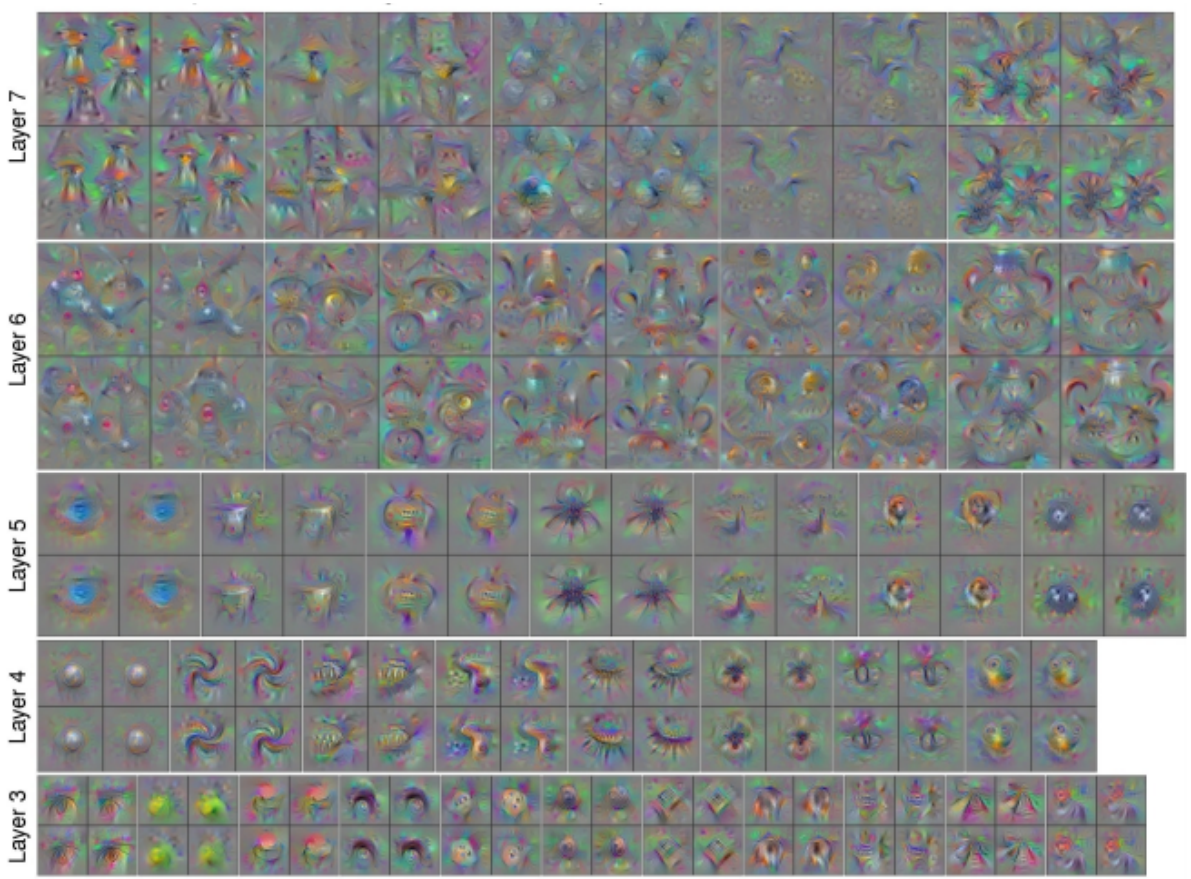


Figure: Visualization of middle nodes

Deep Visualization-5

Figure: Visulization of classification nodes

Deep Visualization-6

Recently, new method called Attention mechanism is introduced.
Intuitively, this mechanism computes 'attention' on each pixel or words.
With Attention, we can visualize reason of classification. [LBH15]

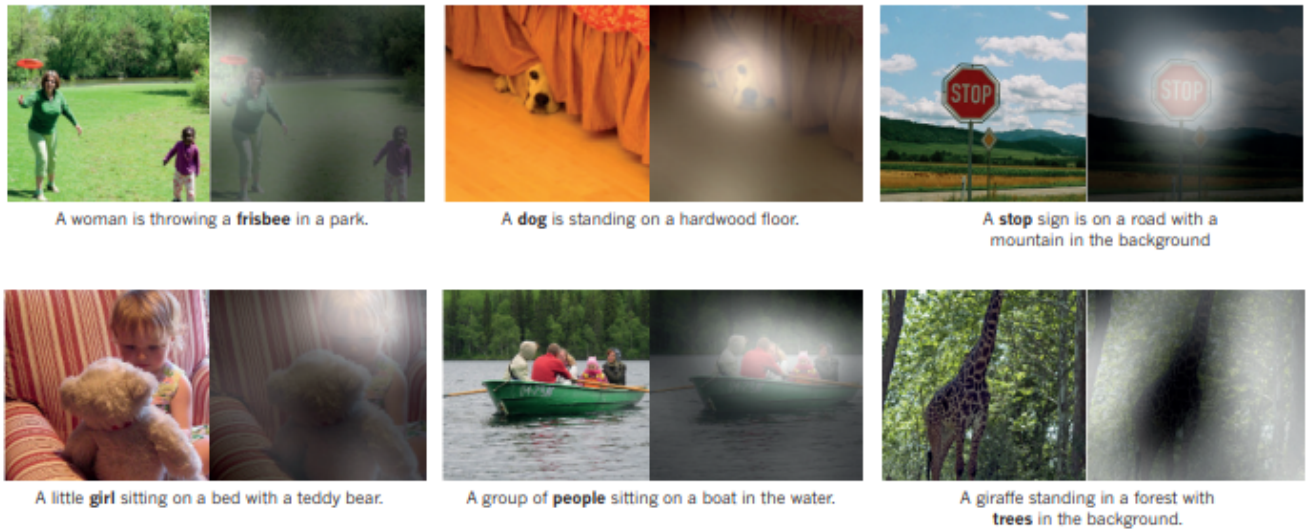


Figure: Image captioning with attention on image

Deep Visualization-7

Attention is used in seq2seq like machine translation, where we can see attention matrix in translation.

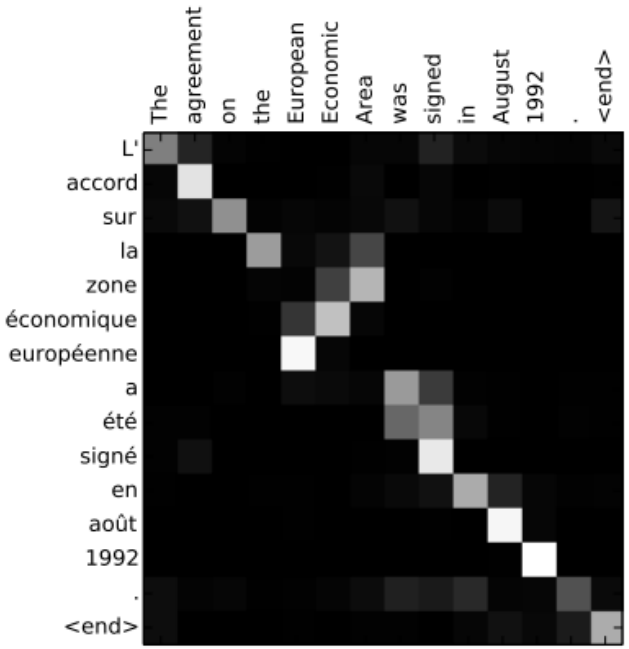


Figure: Attention matrix in machine translation [BCB16]

Deep Visualization-8

- Intuitive
- Not theoretical
- Only appliable to image data
- Attention is applied to various area, including Computer Vision, Natural Language Processing, Program Processing
- But does not explain neural network itself

References I

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

Jason Yosinski et al. *Understanding Neural Networks Through Deep Visualization*. 2015. arXiv: 1506.06579 [cs.CV].

Manifold Hypothesis-1

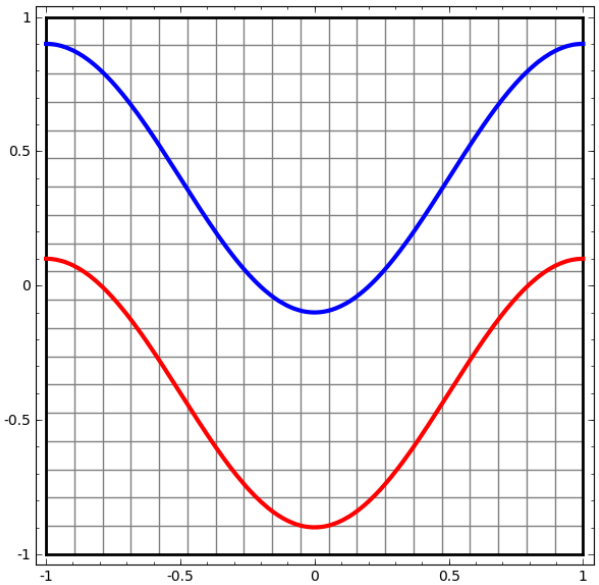
Many deep learning researches relies on hypotheses, like distributional hyopthesis, naturalness hyopthesis, etc.
These hypotheses assumes dataset will behave good, so that designed architecture works well.
Manifold Hypothesis is one of such hypotheses, that dataset forms a (relatively) low dimensional manifold, and neural network learns structure of that manifold.

Definition

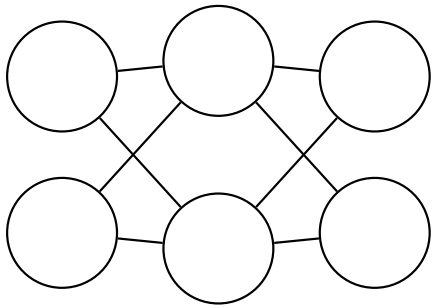
We say some space is n-dimensional manifold if it is locally homeomorphic to $\mathbb{R}^n$. For simplicity, we will not consider specific type of manifolds. You can assume that it is differentiable manifold.

Manifold Hypothesis-2

Let's begin with simple example.

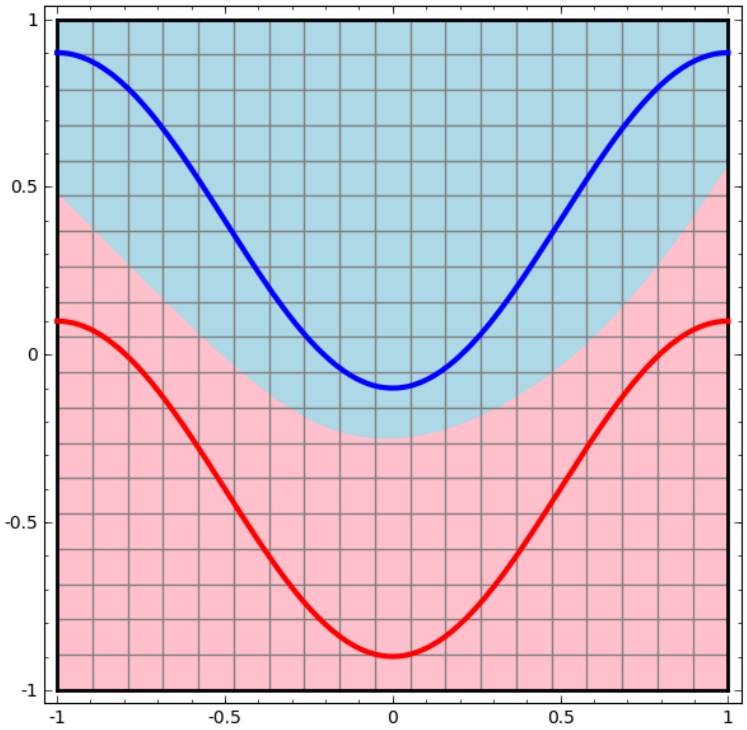


(a) Classification of two one-dimensional manifold [Ola]

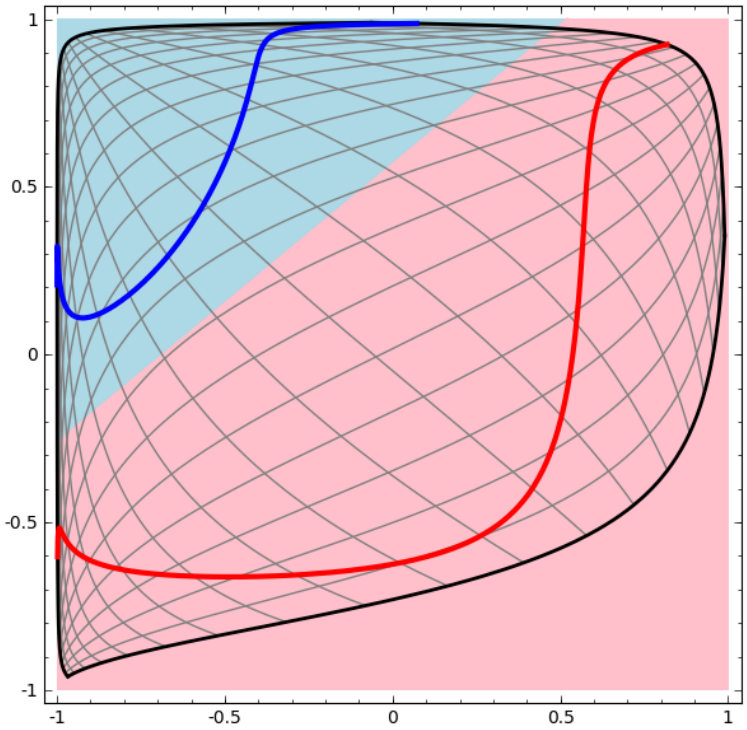


(b) Small neural network used for this problem

Manifold Hypothesis-3



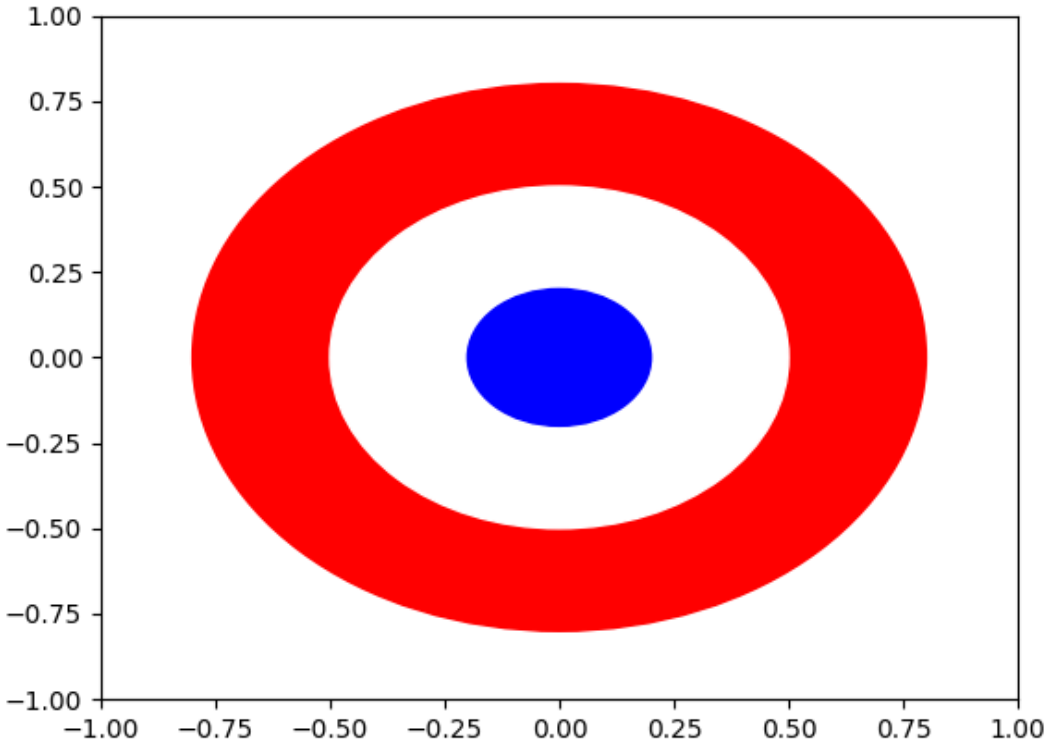
(a) Classification on first layer



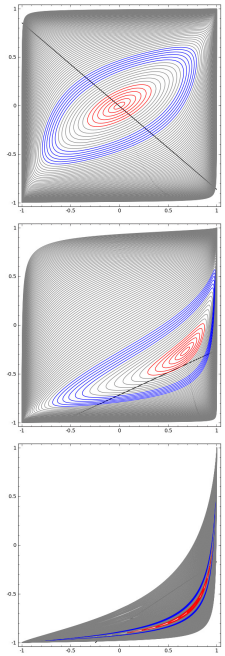
(b) Classification on second layer

Manifold Hypothesis-4

Let's see more complex example.



(a) Data hard to classify



(b) Process of learning

Manifold Hypothesis-5

- Why did preceding example not classified?
- Because they are linked
- Not classfiable by homeomorphism
- But possible in 3D

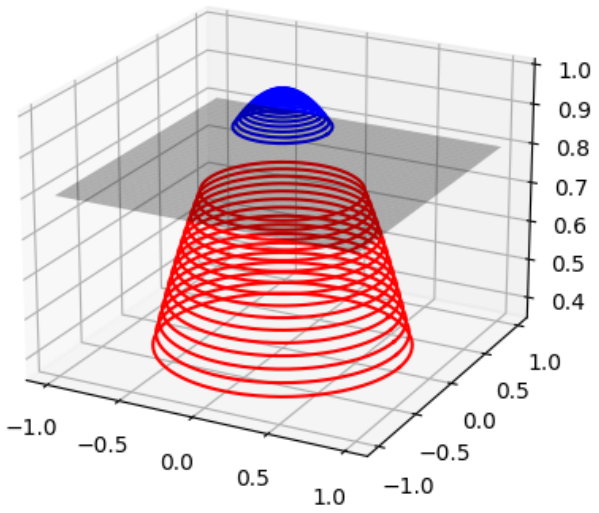


Figure: Classification on 3D

Manifold Hypothesis-6

- Knowing dimension of data-manifold help creating efficient embedding
- Knowing structure of data-manifold help to know width required
- However, this is very complex. Even most simple MNIST dataset is still unknown.

Definition

Intrinsic Dimension is the minimal number of coordinates which are necessary to describe its points without significant information loss.

Manifold Hypothesis-7

Proposition

Let x_i be uniformly sampled on a manifold with intrinsic dimension d and let N be the total number of points. If $r_i^{(1)}$ and $r_i^{(2)}$ are distances of the first and the second neighbor of i-th data, respectively, then

$$\mu_i = \frac{r_i^{(2)}}{r_i^{(1)}} \sim Pareto(d+1)$$

we can compute likelihood of intrinsic dimension as [Ans+19]

$$P(d|\mu) = d^N \prod_{i=1}^{N} \mu_i^{-(d+1)}$$

Manifold Hypothesis-8

One research proposed an algorithm testing the manifold hypothesis, given population P determines which one is true in

- There exists $M \in G(d, CV, \frac{\tau}{C})$ such that $L(M, P) \leq C\epsilon$
- There exists no $M \in G(d, V/C, C\tau)$ such that $L(M, P) \leq \frac{\epsilon}{C}$

with probability at least $1 - \delta$ [FMN13].

References I

Alessio Ansuini et al. "Intrinsic dimension of data representations in deep neural networks". In: *CoRR* abs/1905.12784 (2019). arXiv: 1905.12784. URL: http://arxiv.org/abs/1905.12784.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. *Testing the Manifold Hypothesis*. 2013. arXiv: 1310.0425 [math.ST].

Christopher Olah. *Neural Networks, Manifolds, and Topology*. https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/. Accessed: 2021-04-29.

Finite Approximation-1

Let's consider following problem.
For neural network, what happens if we send each hyperparameters to infinite?

- Width
- Depth
- Number of steps
- Sample size

Finite Approximation-2

Definition

A time continuous stochastic process $\{X_t\}_{t \in T}$ is **Gaussian** if for every finite indices $t_1, \ldots, t_k \in T$

$$X_{t_1,\ldots,t_k} = (X_{t_1}, \ldots, X_{t_k})$$

is a multivariate gaussian random variable.

Theorem

A infinite width with single layer is equivalent to a Gaussian Process. [HJ15]

Finite Approximation-3

A standard feedforward neural network is C^0 function

$$x^{(l+1)} := f(x^{(l)}; l)$$

A residual network has the form

$$x^{(l+1)} := x^{(l)} + f(x^{(l)}; l)$$

where by sending depth L to infinity we achieve

$$x^{(l+1)} \simeq x^{(l)} + f(x^{(l)}; l)\Delta l$$

which is a C^1 coordinate transform.
Similarly, C^k neural network defines a C^k coordinate transform, which becomes solving differential equation of degree k. [HR17]

Finite Approximation-4

Generative Adversarial Network is an architecture that tries to generate data by training two adversarial network. Its loss

$$\mathcal{L}(\theta,\phi) := \mathbb{E}_{x\sim p(x)}[\log(D(x;\theta))] + \mathbb{E}_{z\sim p(z)}[\log(1-D(G(z;\phi);\theta))]$$

which then becomes ordinary differential equation [Qin+20]

$$\begin{pmatrix} \frac{d\theta}{dt} \\ \frac{d\phi}{dt} \end{pmatrix} = -[\alpha\frac{\partial\mathcal{L}_D}{\partial\theta}, \beta\frac{\partial\mathcal{L}_G}{\partial\phi}]$$

References I

Tamir Hazan and Tommi S. Jaakkola. "Steps Toward Deep Kernel Methods from Infinite Neural Networks". In: *CoRR* abs/1508.05133 (2015). arXiv: 1508.05133. URL: http://arxiv.org/abs/1508.05133.

Michael Hauser and Asok Ray. "Principles of Riemannian Geometry in Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/0ebcc77dc72360d0eb8e9504c78d38bd-Paper.pdf.

Chongli Qin et al. *Training Generative Adversarial Networks by Solving Ordinary Differential Equations*. 2020. arXiv: 2010.15040 [stat.ML].

Global Minimum-1

Now suppose we have 'Good' approximation for our desired function.
How can we find it? Moreover, can we find it by gradient descent?

- We may require too many steps.
- We may stuck in local minimum.
- We may stuck in saddle point.

Global Minimum-2

Definition

Gaussian random field is a stochastic process over euclidean space $\{X_x\}_{x\in\mathbb{R}^n}$ that for every finite points $x_1, \ldots, x_k \in \mathbb{R}^n$

$$X_{x_1,\ldots,x_k} = (X_{x_1}, \ldots, X_{x_k})$$

is a multivariate gaussian random variable.

We can approximate loss space of neural network with gaussian random field.

Global Minimum-3

Theorem

For a critical point with error ϵ from the global minimum, are exponentially likely to be saddle points.
Conversely, all local minimum are likely to have an error very close to that of the global minimum. [Dau+14; BD07]
Intuitively in high dimensions the probability that all the partial derivative lead upward is exponentially small w.r.t. the number of dimensions.

And through empirical evaluation, this also holds in neural network's loss space.

Global Minimum-4

We can validate former result with random matrix theory.
According to Wigner's semicircular law, eigenvalue distribution of large Gaussian random matrix follows semicircle distribution with mean 0.
Hessian matrix of loss space behaves similar, but shifted by an amound determined by $\epsilon/$

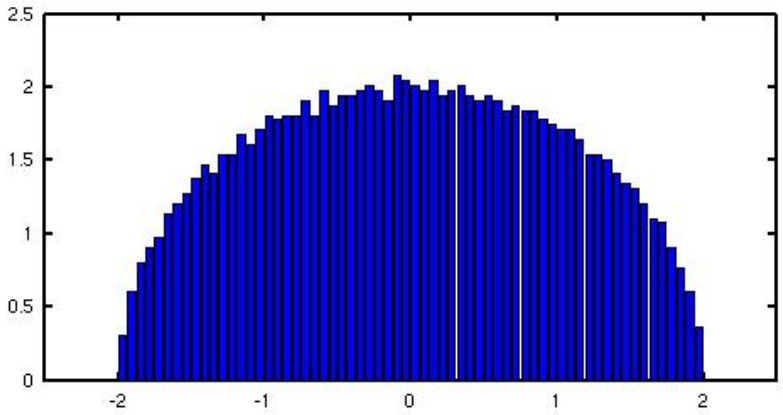


Figure: Wigner's semicircular law

Global Minimum-5

- For global minimum, distribution is shifted right so that all eigen values are positive.
- As ϵ increases, it includes more negative eigenvalues with high probability of zero eigenvalues.
- This indicates existence of plateau around saddle point with large error.
- So if we can escape plateau, we can escape saddle point.

Global Minimum-6

There are research that analyzed behavior of local minimums in neural network. [HWT20]

1. The loss surface of neural network with piecewise linear activation is partitioned into multiple smooth open cells.
2. If we have single hidden layer with two-piece linear activation and convex loss, all local minima are global minimas in cells.
3. Moreover if we have strictly convex loss, all the local minima in each cells are path-connected.

Global Minimum-7

Moreover, gradient descent 'can' find global minimum. Moreover, we can always find neural network.

Theorem

For three networks, fully-connected feedforward network, ResNet, convolutional ResNet with enough width $m(n, \delta, K^{(H)}, \lambda_{min}, H)$ *randomly initialized gradient descent satisfies following inequality with probability at least* $1 - \delta$. *[Du+18]*

$$L(\theta(k)) \leq \left(1 - \frac{\eta \lambda_{min}(K^{(H)})}{2}\right)^k L(\theta(0))$$

References I

Alan J. Bray and David S. Dean. "Statistics of Critical Points of Gaussian Fields on Large-Dimensional Spaces". In: *Physical Review Letters* 98.15 (Apr. 2007). ISSN: 1079-7114. DOI: 10.1103/physrevlett.98.150201. URL: http://dx.doi.org/10.1103/PhysRevLett.98.150201.

Yann N Dauphin et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/17e23e50bedc63b4095e3d8204ce063b-Paper.pdf.

Simon S. Du et al. "Gradient Descent Finds Global Minima of Deep Neural Networks". In: *CoRR* abs/1811.03804 (2018). arXiv: 1811.03804. URL: http://arxiv.org/abs/1811.03804.

Fengxiang He, Bohan Wang, and Dacheng Tao. *Piecewise linear activations substantially shape the loss surfaces of neural networks*. 2020. arXiv: 2003.12236 [cs.LG].

Generalizability-1

So far we checked

- There exists a neural network that approximates given function.
- Trained neural network shows reasons that they learn function well.
- And we can find global minimum with gradient descent.

But the problem is, loss zero does not means that we have 100% accuracy. Neural network has to generalize its learning.

Generalizability-2

We say a model is **overfitted** when it learns training dataset too closely, so that it may fails to fit other data or predict future observations.
To prevent overfitting, we use regularization, dropout, cross validation, etc.

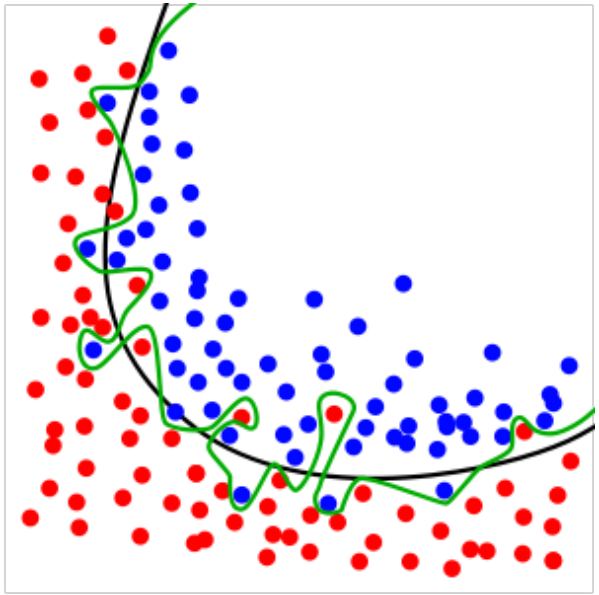


Figure: Black line represents reasonable classifier, where green line represents overfitted one. [Wik21]

Generalizability-3

Neural network has extreme over parametrization, since it contains enormous number of parameters.
One of famous research have shown that neural network can learn all following corrupted data. [Zha+16]

- All labels are changed randomly.
- Pixels are shuffled universally.
- Pixels are shuffled individually.
- Pixels are sampled from Gaussian distribution.

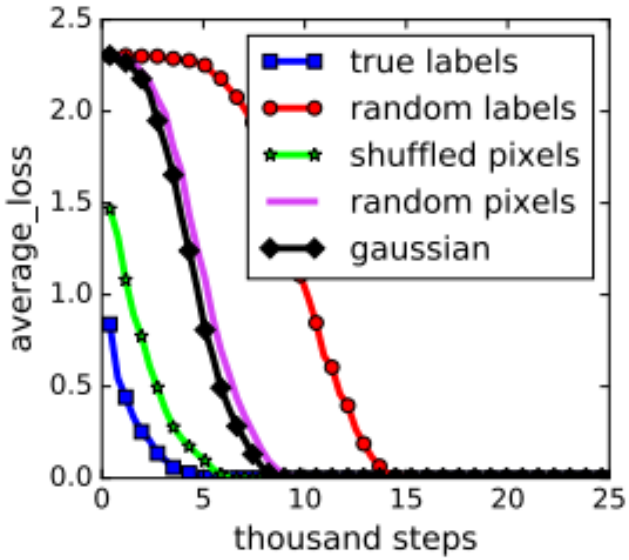


Figure: Risk curves for corrupted data

Generalizability-4

Usually machine learning's risk curve w.r.t. model capacity looks like right one.
Recent research argues that it is actually a double descent risk curve. [Bel+19]
This is empirically tested with RFF model, fully connected neural network, random forests.

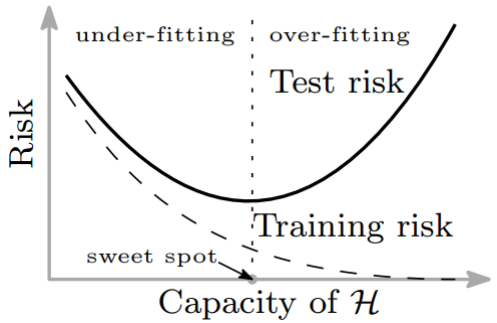


Figure: Classical Learning Curve

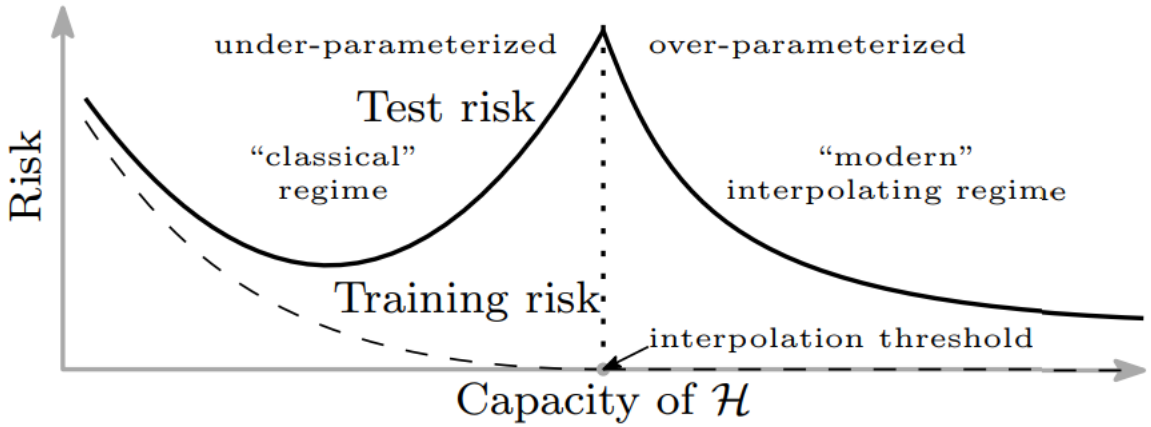


Figure: Modern Learning Curve

Generalizability-5

We can define generalization gap as

$$\mathcal{R}[f_{A(S)}] - \mathcal{R}_S[f_{A(S)}]$$

where S is samples, $f_{A(S)}$ is a function that algorithm A makes given sample S, $\mathcal{R}$ and $\mathcal{R}_S$ is a expected risk and empirical risk, resp.
Our goal is to find upper bound of generalization gap for neural network, with

$$\mathcal{R}[f_{A(S)}] - \mathcal{R}_S[f_{A(S)}] \leq \sup_{f \in F} \mathcal{R}[f] - \mathcal{R}_S[f] = \mathcal{M}$$

Generalizability-6

Here I list some of bounds that holds with probability at least $1-\delta$.

$$\mathcal{M} \leq 2\mathfrak{R}_m(\mathcal{L}_f) + \sqrt{\frac{\log 1/\delta}{2m}}$$

where $\mathfrak{R}_m(\mathcal{L}_f)$ is the Rademacher complexity.

$$\mathcal{M} \leq 2\beta + (4m\beta + M)\sqrt{\frac{\log 1/\delta}{2m}}$$

where β is a uniform stability.

$$|\mathcal{M}| \leq \zeta(S) + M\sqrt{\frac{2\Omega \log 2 + 2\log 1/\delta}{m}}$$

where algorithm is $(\Omega, \zeta(\cdot))$-robust. [KKB20]

Generalizability-7

Another comes from observation that the more complex the model is, it has less generalizability.
So by analyzing complexity of decision boundary, we can measure generalizability. [GL20]

By creating adversarial set as following, we can sample from decision boundary.
Then applying PCA to adversarial set, and compute Shannon entropy of eigen values.

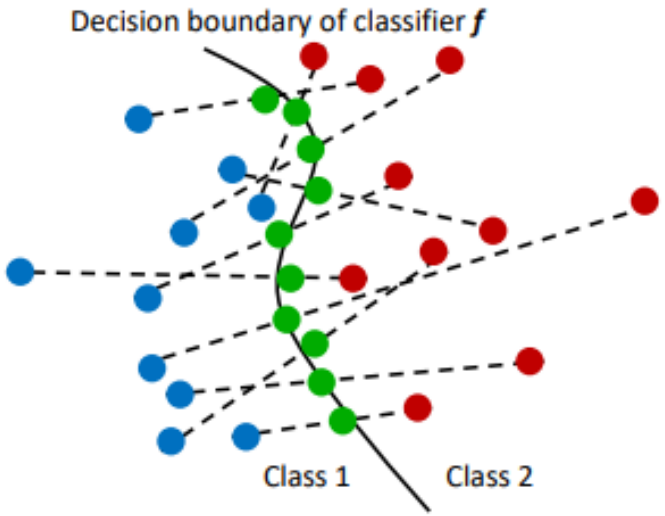


Figure: Findinf decision boundary by interpolation

References I

Mikhail Belkin et al. *Reconciling modern machine learning practice and the bias-variance trade-off*. 2019. arXiv: 1812.11118 [stat.ML].

Shuyue Guan and Murray Loew. "Analysis of Generalizability of Deep Neural Networks Based on the Complexity of Decision Boundary". In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)* (Dec. 2020). DOI: 10.1109/icmla51294.2020.00025. URL: http://dx.doi.org/10.1109/ICMLA51294.2020.00025.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. *Generalization in Deep Learning*. 2020. arXiv: 1710.05468 [stat.ML].

Wikipedia contributors. *Overfitting — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-April-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Overfitting&oldid=1016721642.

Chiyuan Zhang et al. "Understanding deep learning requires rethinking generalization". In: *CoRR* abs/1611.03530 (2016). arXiv: 1611.03530. URL: http://arxiv.org/abs/1611.03530.

Verification-1

In programming language theory, we verify that programs are correct with formal methods. For formal methods, we include

- Proof Assistanct
- Static Analysis
- Model checking

We don't want following adversarial examples not happen.

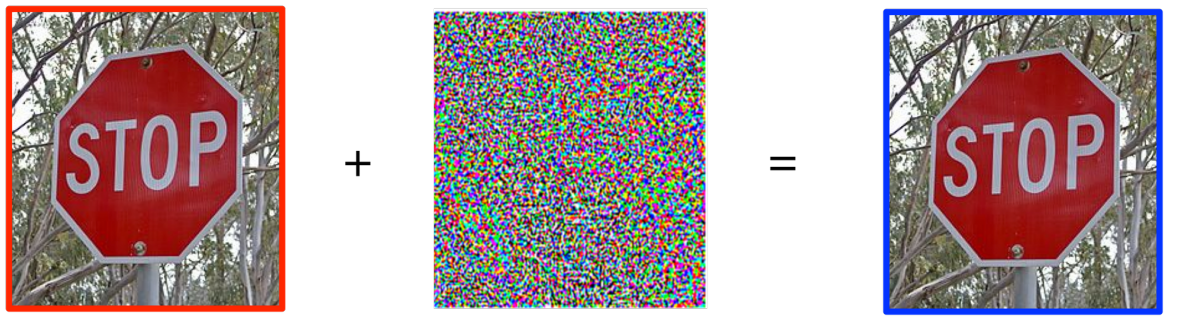


Figure: Adversariable example misclassified [Jie19]

Verification-2

Usually our goal is to verify that there is no adversarial example.

Definition

We say a classifier F is δ-locally-robust at data point x if for every x' that $||x - x'||_\infty \leq \delta$ we have $F(x) = F(x')$.
For regression function F, we can use Lipscitz constant L which satisfies following

$$||F(x) - F(x')|| \leq L||x - x'||$$

Verification-3

The first idea is simple. We 'solve' the linear equation that neural network defines.
For each variable, we only record its maximum/minimum values.
Since each layer is linear, and ReLU is piecewise linear, we can safely track maximum/minimum values.
By feedforwarding initial square $\{x' : ||x - x'||_\infty \leq \delta\}$ through neural network, we can check whether they have same label.
This procedure is automated with SMT solver. [Kat+17]

Verification-4

Definition

Abstract Interpretation is a family of sets on $\mathbb{R}^n$, that is used to over-approximation, which(can be used to statically check numerical problem.
For some function F, if we have overapproximation ($X \subseteq A$)of input set X by AI A, we can feed forward A into F to overapproximate $F(X) \subseteq \bar{F}(A)$.

We can use various abstract interpretation for neural network verification, like simplex, zonotope, polyhedra.
Simplex was not enough to verify all network, and polyhedra took many days even in simplest network. [Geh+18]

Verification-5

Following theorem gives another approach. [RHK18]

Theorem

Suppose we need to minimize $w(x)$ for $x \in [a,b]^n$.
If w is Lipscitz continuous, there exists some function h that

$$\forall x, y \in [a,b]^n, h(x,y) \leq w(x) \text{ and } h(x,x) = w(x)$$

Moreover, there is sequence of Y_i that $Y_i \subset Y_j$ for $i < j$ with

$$\inf_{x \in [a,b]^n} \max_{y \in Y_i} h(x,y) \to \inf_{x \in [a,b]^n} w(x)$$

Since neural network is Lipscitz continuous with operator norm of linear layers, we can approximate minimum/maximum value of neural network with sequence.

Verification-6

Another way is approximating the neural network, and verify the approximation. This research verifies reinfocement learning by guarding. [Zhu+19]

1. Approximate neural network with piecewise linear function P_θ.
2. Find invariant Φ that
   - Every initial state is in Φ
   - No danger state is in Φ
   - If $s \in \Phi$, then $P_\theta(s) \in \Phi$
3. Now when using neural network, if its step violates Φ, we use P_θ instead.

References I

Timon Gehr et al. "AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation". In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 3–18. DOI: 10.1109/SP.2018.00058.

Xi Wu Jiefeng Chen. *Robust Attribution Regularization*. [Online; accessed 30-April-2021]. 2019. URL: https://deep.ghost.io/robust-attribution/.

Guy Katz et al. "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks". In: *CoRR* abs/1702.01135 (2017). arXiv: 1702.01135. URL: http://arxiv.org/abs/1702.01135.

Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. *Reachability Analysis of Deep Neural Networks with Provable Guarantees*. 2018. arXiv: 1805.02242 [cs.LG].

He Zhu et al. "An Inductive Synthesis Framework for Verifiable Reinforcement Learning". In: *CoRR* abs/1907.07273 (2019). arXiv: 1907.07273. URL: http://arxiv.org/abs/1907.07273.

Testing-1

In software engineering, we create test cases to check whether software is correct.
We usually check coverage as measure how test cases tests well, like statement coverage, condition coverage, mutation coverage, etc.
We'd like to define coverage for neural network, and find good test examples.

Testing-2

Recall the definition of activation value.

$$x_{i,j} = \sigma(\sum_{k=1}^{N_{i-1}} w_{k,j} x_{i-1,k} + b_{i,j})$$

We say a neuron is covered if its activation value is larger than threshold, $x_{i,j} > \epsilon$, like 0.3. [Pei+17]
Given n Neural Newtorks $NN_1, \ldots, NN_n$, our goal is to find data that they disagree and and neuron coverage is large enough by maximizing following loss.

$$\sum_{i \neq j} NN_i(x)[c] - \lambda_1 NN_j(x)[c] + \lambda_2 \alpha_{i,j}(x)$$

Testing-3

There are other notions of coverage

- k-multisection Neuron Coverage [Ma+18]
- Neuron Boundary Coverage [Ma+18]
- Strong Neuron Activation Coverage [Ma+18]
- Sign(Value)-Sign(Value) Coverage [SHK18]
- Surprise Adequacy [KFY18]

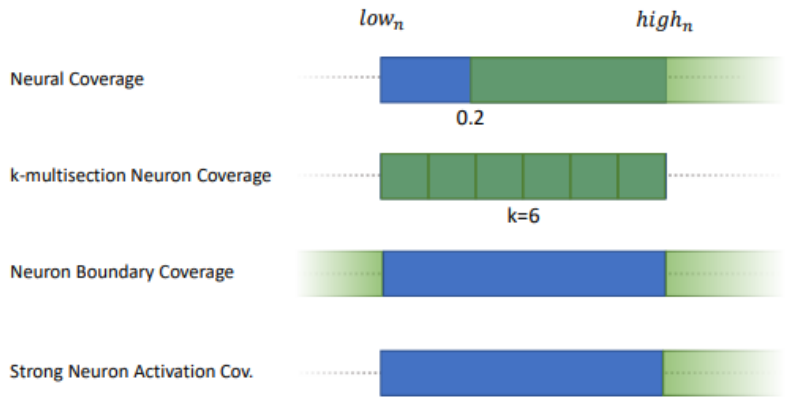


Figure: Visualization of neuron coverages

References I

Jinhan Kim, Robert Feldt, and Shin Yoo. "Guiding Deep Learning System Testing using Surprise Adequacy". In: *CoRR* abs/1808.08444 (2018). arXiv: 1808.08444. URL: http://arxiv.org/abs/1808.08444.

Lei Ma et al. "DeepGauge: Comprehensive and Multi-Granularity Testing Criteria for Gauging the Robustness of Deep Learning Systems". In: *CoRR* abs/1803.07519 (2018). arXiv: 1803.07519. URL: http://arxiv.org/abs/1803.07519.

Kexin Pei et al. "DeepXplore: Automated Whitebox Testing of Deep Learning Systems". In: *CoRR* abs/1705.06640 (2017). arXiv: 1705.06640. URL: http://arxiv.org/abs/1705.06640.

Youcheng Sun, Xiaowei Huang, and Daniel Kroening. "Testing Deep Neural Networks". In: *CoRR* abs/1803.04792 (2018). arXiv: 1803.04792. URL: http://arxiv.org/abs/1803.04792.

Throughout the survey, following surveys helped a lot on finding papers. [HT21; Gui+18; UM21; 18]
Large portion of papers are accepted to following conferences, including ICML, ICLR, NIPS, IJCAI.

"Safety and Trustworthiness of Deep Neural Networks: A Survey". In: *CoRR* abs/1812.08342 (2018). Withdrawn. arXiv: 1812.08342. URL: http://arxiv.org/abs/1812.08342.

Riccardo Guidotti et al. "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* 51.5 (Aug. 2018). ISSN: 0360-0300. DOI: 10.1145/3236009. URL: https://doi.org/10.1145/3236009.

Fengxiang He and Dacheng Tao. *Recent advances in deep learning theory*. 2021. arXiv: 2012.10931 [cs.LG].

Caterina Urban and Antoine Miné. *A Review of Formal Methods applied to Machine Learning*. 2021. arXiv: 2104.02466 [cs.PL].