Foundations of Deep Learning overview

16 August 2019

In this talk

Will be covered

- nonparametric supervised offline regression

Will NOT be covered

- unsupervised
- online
- density estimation
- reinforcement learning
- beyond i.i.d
- generative models
- adversarial learning
- computational complexity
- approximation power

Plan

Generalization [1]–[4]

$$Z = (X, Y) \text{ — r.v.} \quad X \in \mathcal{X}, Y \in \mathcal{Y}$$

loss $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

population risk $L : \mathfrak{M}(\mathcal{Y}^{\mathcal{X}}) \to \mathbb{R}$

$$L(f) = \mathbb{E}_Z[\ell(f(X), Y)]$$

Aim:

$$f^\star = \arg\min_{f \in \mathfrak{M}(\mathcal{Y}^{\mathcal{X}})} L(f)$$

Simplification: level 1.a

Issue

$\mathfrak{M}(\mathcal{Y}^{\mathcal{X}})$ is too large, not parametrizable

Solution

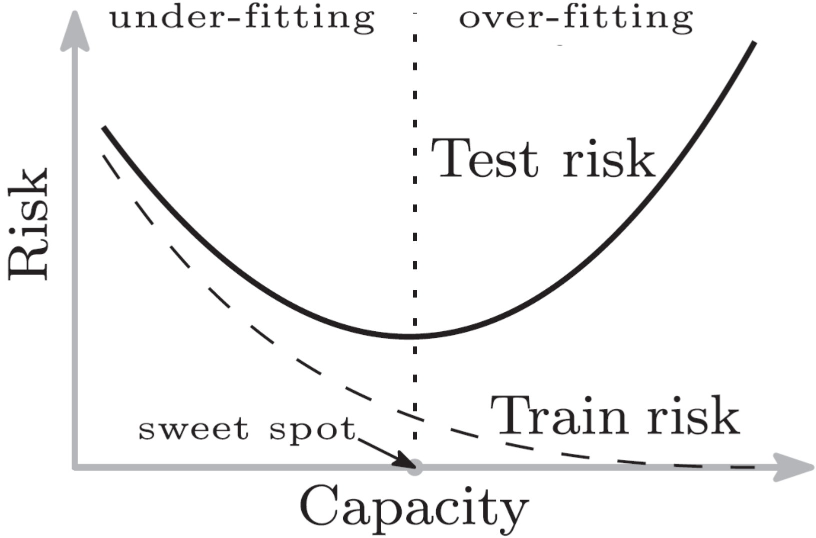
Introduce $\mathcal{F} \subset \mathfrak{M}(\mathcal{Y}^{\mathcal{X}})$ — parametric function class, incapsulates inductive bias

bias-variance trade-off

for $f_{\mathcal{F}} = \arg\min_{f \in \mathcal{F}} L(f)$, any $f \in \mathcal{F}$:

$$\underbrace{L(f) - L(f^{\star})}_{\text{total risk}} = \underbrace{L(f) - L(f_{\mathcal{F}})}_{\text{estimation risk}} + \underbrace{L(f_{\mathcal{F}}) - L(f^{\star})}_{\text{approximation risk}}$$

Classic risk curve



Simplification: level 1.b

Issue

L is defined with unknown $\mathbb{P}_Z$

Solution

Sample i.i.d $(X_i', Y_i')_{i=1}^m$ from $\mathbb{P}_Z$ and define empirical risk

$$\tilde{L}(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(f(X_i'), Y_i')$$

Law of Large Numbers

If $f \perp\!\!\!\perp (X_i', Y_i')^m$ then $\tilde{L}(f) - L(f) \to 0$

Inconsistency

Aim:

$$\tilde{f} = \arg\min_{f \in \mathcal{F}} \tilde{L}(f)$$

$\tilde{f}$ defined by stochastic objective $\tilde{L}(f)$ that depends on $(X_i', Y_i')^m$, but $\tilde{f}$ must not depend on $(X_i', Y_i')^m$ to use LLN

Simplification: level 2

Issue

$\tilde{f}$ depends on $(X_i', Y_i')^m$

Solution

Generate a copy: i.i.d $(X_i, Y_i)_{i=1}^n$ from $\mathbb{P}_Z$ and build

$$\hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(X_i), Y_i) \qquad \hat{f} = \arg\min_{f \in \mathcal{F}} \hat{L}(f)$$

Simplification: level 3

Issue

$\hat{L}(f)$ can still be non-convex, NP hard to optimize

Solution

fix a learning algorithm $\mathcal{A}$ and take

$$\hat{f} = \mathcal{A}(\{Z_i\}^n, \Gamma)$$

Use intrinsic properties of the algorithm to build generalization guarantees

Power system example

Power system state estimation problem via non-convex recovery:

$$\text{given } X_i = M_i \text{ and } Y_i = m_i$$

$$\text{minimize} \sum_i (x^* M_i x - m_i)^2$$

Power engineering aim: get $\|xx^* - vv^*\|$ small
Machine Learning aim: get $\mathbb{E}_{M,m}[x^* M x - m]^2$ small

Statistical Learning Theory

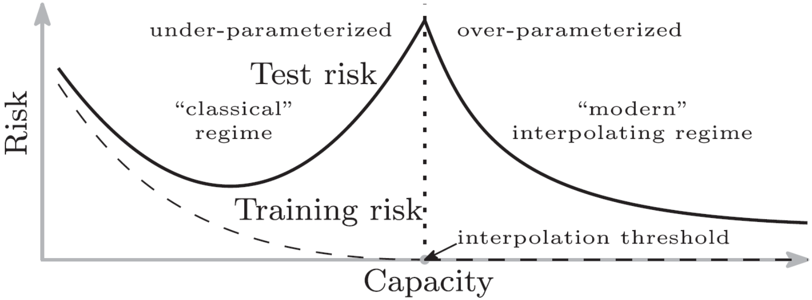
Bounds on Excess Risk

$$L(f) - L(f_{\mathcal{F}}) = [L(f) - \hat{L}(f)] + [\hat{L}(f) - \hat{L}(f_{\mathcal{F}})] + [\hat{L}(f_{\mathcal{F}}) - L(f_{\mathcal{F}})]$$

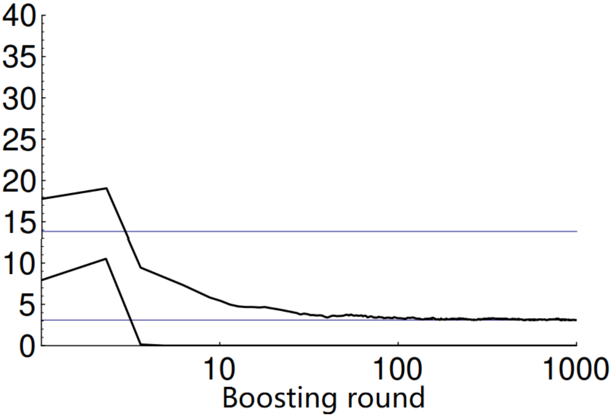
$$\mathbb{E}_n L(\hat{f}) - L(f_{\mathcal{F}}) \leq \mathbb{E}_n[L(\hat{f}) - \hat{L}(\hat{f})] \leq \mathbb{E}_n \sup_{f \in \mathcal{F}} |L(f) - \hat{L}(f)|$$

Note: exist bounds with compression argument and stability argument, which do not rely on properties of $\mathcal{F}$, but on $\hat{f}$ instead

Modern risk curve [5]

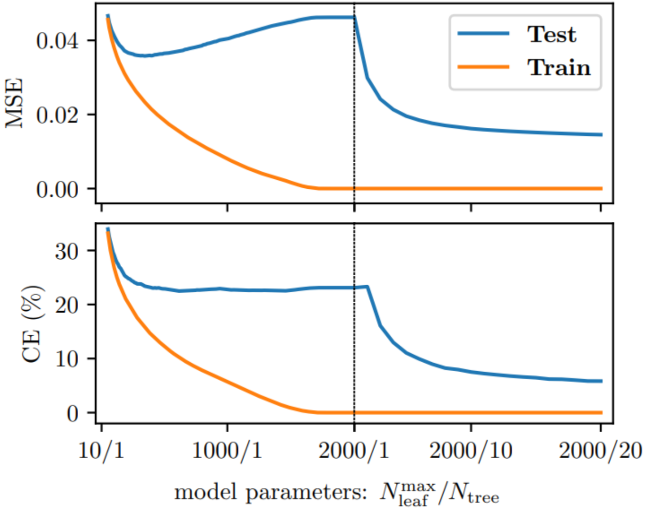


"Modern" risk curve

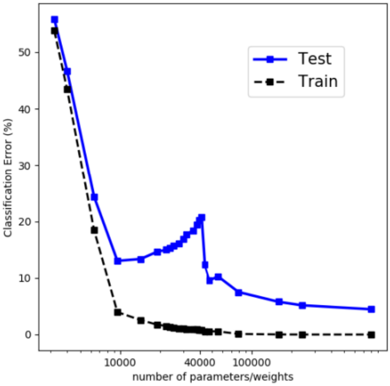


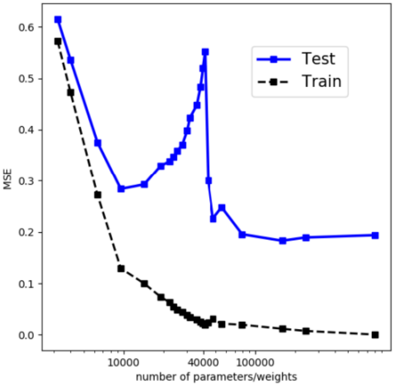
NIPS tutorial by P.Bartlett, 1998

More examples: random forest [5]



More examples: neural net [5]





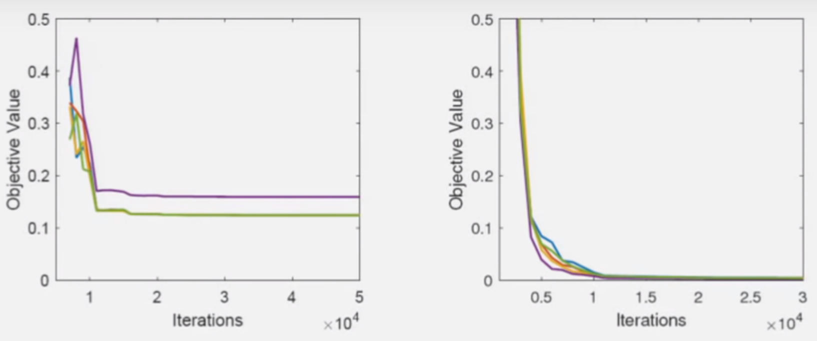
Questions

Phenomenon: good test risk with zero *regression* train loss on *noisy* data

1. When is zero train loss achievable? (optimization)
2. In which of these cases does test risk go down? (generalization)
3. What is the gain compared to the amount of computation?

State of the art: optimization [7], [8]

Observation: larger architectures are easier to train [6]



There is a phase transition

State of the art: optimization [7], [8]

Taxometry of results on overparametrization

- Static: All solutions (SOSP) are global
  - Shallow, quadratic activation: $\textit{width} \gtrsim \min\{\textit{dim}, \sqrt{n}\}$ [9], [10]
  - Deep, leaky ReLU: $\textit{width} \gtrsim n$ [11], [12]
- Dynamic: SGD finds global solution near a random initialization (SGD learns something competitive to the best in RKHS) [13]–[16]
  - $\textit{width} \gtrsim n^2$ at least and depth-dependent

Note: under regularity conditions, SGD converges to a second-order stationary point [17], [18]

Reproducing Kernel Hilbert Space (RKHS)

$\mathcal{H}$ — Hilbert (complete inner product vector) space of real-valued functions on $\mathcal{X}$. It is a RKHS iff there is $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that

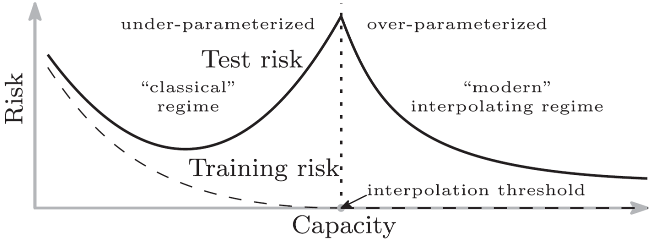
$$\sum_{i,j} c_i c_j K(x_i, x_j) \geq 0$$

$$\mathcal{H} \longleftrightarrow K$$

Usually, $K(x, y) = \langle \phi(x), \phi(y) \rangle$ where $\phi : \mathcal{X} \to \mathcal{V}$

Examples: Gaussian $e^{-\frac{\|x-y\|^2}{\sigma}}$; Neural Tangent Kernel (NTK) [19], [20]

State of the art: generalization



What happens after interpolation threshold: implicit regularization

- by $\|W_1\| \dots \|W_L\|$ for linear neural nets [21]
- by $\|f\|_{\mathcal{H}}$ for Random Fourier Features [5], [22]

Note: [23] characterizes the effect of overparametrization on generalization in linear regression

Neural Network learning routine [24]–[27]

Steps:

- Model Selection
- Initialization
- Learning algorithm
- Regularization

Routine:

1. reach interpolation
2. regularize

Model selection: Neural architecture search [40]

- Adjustable parameters: layer structure (dense, convolutional, recurrent, attention), activation function, learning properties
- Hierarchical search space [28], [29]
- Search strategy: Evolution [30], [31], Bayessian optimization [32], [33], Reinforcement Learning [34], continuous relaxation [35], incremental learning [36]

- First benchmark dated by May 2019 [37]
- Modern NN models seem to tolerate adaptation to common data sets [38], [39]

Initialization: random

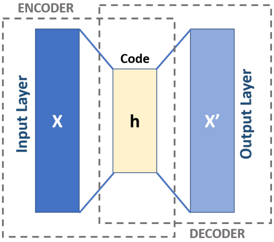
$$\text{Xavier: } W^{(l)} \sim \mathcal{N}(0, \frac{2}{p^{l-1}}); \quad b^{(l)} = 0 \quad [41]$$

$$\text{He: } W^{(l)} \sim \mathcal{N}(0, \frac{2}{p^{l-1} + p^{l}}); \quad b^{(l)} = 0 \quad [42]$$

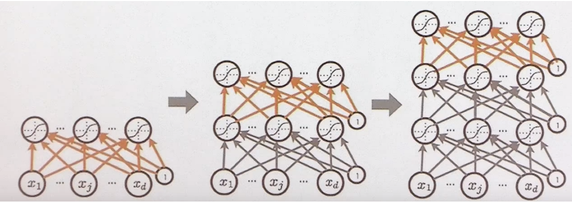
Initialization: Unsupervised pre-training [43]

Autoencoder

$$\phi : \mathcal{X} \to \mathcal{Y}$$
$$\psi : \mathcal{Y} \to \mathcal{X}$$
$$\phi^*, \psi^* = \arg\min_{\phi,\psi} \mathbb{E}\ell[(\psi \circ \phi)(X), X]$$



Greedy layer-wise pre-training: stack encoders

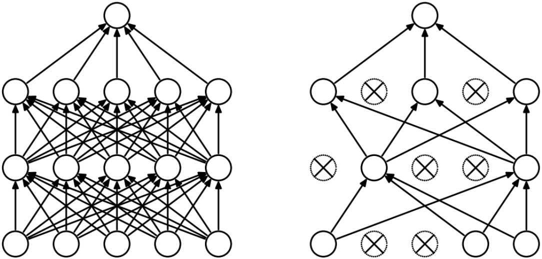


Learning algorithm: SGD

Objective: $\min_\theta \frac{1}{n}\sum_{i=1}^{n} \ell(\Phi(X_i, \theta), Y_i)$
mini-batch SGD: $\theta^{t+1} = \theta^t - \frac{\alpha}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla_\theta \ell(\Phi(X_i, \theta^t), Y_i)$
Modern version: faster convergence/saddles fighting

- Momentum[44]:
  $\theta^{t+1} = \theta^t - v^{t+1}; v^{t+1} = \gamma v^t + \frac{\alpha}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla_\theta \ell(\Phi(X_i, \theta^t), Y_i)$
- Nesterov[45]: in the above $\Phi(X_i, \theta^t) \to \Phi(X_i, \theta^t - \gamma v^t)$
- Adagrad/Adadelta/RMSprop — adaptive learning rate (keep exponentially decaying average of past gradients)
- Batch normalization: $\hat{x}^i = \alpha \frac{x^i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \varepsilon}} + \beta$; $\sigma_\mathcal{B}$ and $\mu_\mathcal{B}$ — std div and mean of the batch, α and β to be learned

Regularization: Dropout [48]



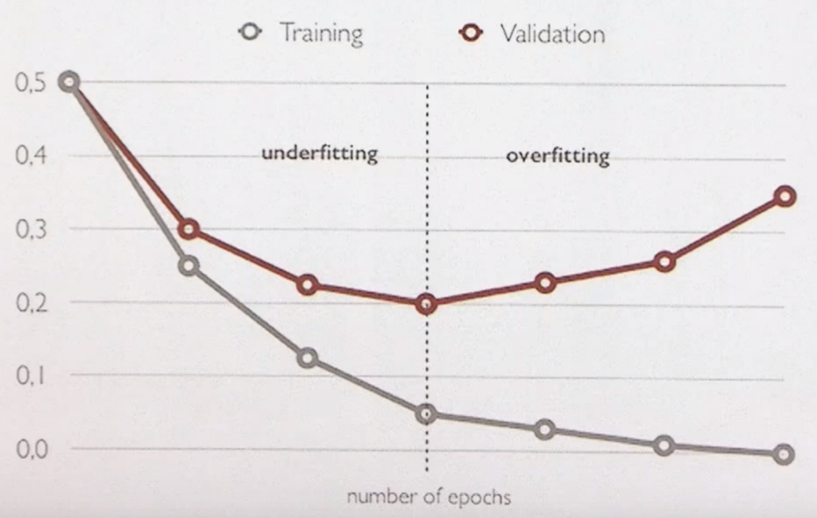
(a) Standard Neural Net (b) After applying dropout.

$z^t \in \{0,1\}^p$ — random, p — number of parameters

SGD+dropout: $\theta^{t+1} = \theta^t - \frac{\alpha}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_\theta \ell(\underbrace{\Phi(X_i, \theta^t, z^t)}_{\text{output of dropout neurons is 0}}, Y_i) \underbrace{\otimes z^t}_{\text{gradient of dropout is 0}}$

Note: "Dropout-stable" solutions posses the property of Landscape Connectivity [46]; dropout acts as explicit regularization in Stochastic Matrix Factorization [47]

Regularization: Early stopping



[1] S. Rakhlin, **Generalization i**, 2019. [Online]. Available: https://youtu.be/Ntl_WNW8yGc.

[2] P. Bartlett, **Generalization ii**, 2019. [Online]. Available: https://youtu.be/QO8HOqQ_6BQ.

[3] S. Rakhlin, **Generalization iii**, 2019. [Online]. Available: https://youtu.be/8hZD5bMBbY4.

[4] P. Bartlett, **Generalization iv**, 2019. [Online]. Available: https://youtu.be/n5Zxi22801Q.

[5] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off", **Proceedings of the National Academy of Sciences**, vol. 116, no. 32, pp. 15 849–15 854, 2019.

[6] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks", in **Advances in neural information processing systems**, 2014, pp. 855–863.

[7] J. Lee, **Recent developments in over-parametrized neural networks, part i**, 2019. [Online]. Available: https://youtu.be/uC2IGoTE2u4.

[8] ——, **Recent developments in over-parametrized neural networks, part ii**, 2019. [Online]. Available: https://youtu.be/NGon2JyjO6Y.

[9] S. S. Du and J. D. Lee, “On the power of over-parametrization in neural networks with quadratic activation”, **arXiv preprint arXiv:1803.01206**, 2018.

[10] R. Ge, J. D. Lee, and T. Ma, “Learning one-hidden-layer neural networks with landscape design”, **arXiv preprint arXiv:1711.00501**, 2017.

[11] Q. Nguyen, M. C. Mukkamala, and M. Hein, “On the loss landscape of a class of deep neural networks with no bad local valleys”, **arXiv preprint arXiv:1809.10749**, 2018.

[12] Q. Nguyen, “On connected sublevel sets in deep learning”, **arXiv preprint arXiv:1901.07417**, 2019.

[13] S. Oymak and M. Soltanolkotabi, "Towards moderate overparameterization: Global convergence guarantees for training shallow neural networks", **arXiv preprint arXiv:1902.04674**, 2019.

[14] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization", **arXiv preprint arXiv:1811.03962**, 2018.

[15] Y. Cao and Q. Gu, "Generalization bounds of stochastic gradient descent for wide and deep neural networks", **arXiv preprint arXiv:1905.13210**, 2019.

[16] S. Arora, N. Cohen, N. Golowich, and W. Hu, "A convergence analysis of gradient descent for deep linear neural networks", **arXiv preprint arXiv:1810.02281**, 2018.

[17] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points—online stochastic gradient for tensor decomposition", in **Conference on Learning Theory**, 2015, pp. 797–842.

[18] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, "Gradient descent only converges to minimizers", in **Conference on learning theory**, 2016, pp. 1246–1257.

[19] S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang, "On exact computation with an infinitely wide neural net", **arXiv preprint arXiv:1904.11955**, 2019.

[20] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks", in **Advances in neural information processing systems**, 2018, pp. 8571–8580.

[21] R. Vidal, **Mathematics of deep learning**, 2018. [Online]. Available: https://youtu.be/yEVphdvn06M.

[22] M. Belkin, **From classical statistics to modern machine learning**, 2019. [Online]. Available: https://youtu.be/OBCciGnOJVs.

[23] P. Bartlett, **Benign overfitting in linear prediction**, 2019. [Online]. Available: https://youtu.be/GXpP-rXEDpk.

[24] R. Salakhutdinov, **Tutorial on deep learning i**, 2017. [Online]. Available: https://youtu.be/-SY4-GkDM8g.

[25] ——, **Tutorial on deep learning ii**, 2017. [Online]. Available: https://youtu.be/R_o6nUC1Nzo.

[26] ——, **Tutorial on deep learning iii**, 2017. [Online]. Available: https://youtu.be/jompe29_A74.

[27] ——, **Tutorial on deep learning iv**, 2017. [Online]. Available: https://youtu.be/SNKypQ8y6SM.

[28] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation", in **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**, 2018, pp. 2423–2432.

[29] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search", **arXiv preprint arXiv:1711.00436**, 2017.

[30] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent network architectures”, in **International Conference on Machine Learning**, 2015, pp. 2342–2350.

[31] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, “Designing neural networks through neuroevolution”, **Nature Machine Intelligence**, vol. 1, no. 1, pp. 24–35, 2019.

[32] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, “Taking the human out of the loop: A review of bayesian optimization”, **Proceedings of the IEEE**, vol. 104, no. 1, pp. 148–175, 2015.

[33] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, “Towards automatically-tuned neural networks”, in **Workshop on Automatic Machine Learning**, 2016, pp. 58–65.

[34] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning”, **arXiv preprint arXiv:1611.01578**, 2016.

[35] H. Liu, K. Simonyan, and Y. Yang, “Darts: Differentiable architecture search”, **arXiv preprint arXiv:1806.09055**, 2018.

[36] Q. Liu, **Splitting gradient descent for incremental learning of neural architectures**, 2019. [Online]. Available: `https://youtu.be/5LMXNZdJ14k`.

[37] C. Ying, A. Klein, E. Real, E. Christiansen, K. Murphy, and F. Hutter, “Nas-bench-101: Towards reproducible neural architecture search”, **arXiv preprint arXiv:1902.09635**, 2019.

[38] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, “Do cifar-10 classifiers generalize to cifar-10?”, **arXiv preprint arXiv:1806.00451**, 2018.

[39] B. Recht, **Training on the test set and other heresies**, 2019. [Online]. Available: `https://youtu.be/NTz4rJS9BAI`.

[40] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey", **arXiv preprint arXiv:1808.05377**, 2018.

[41] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in **Proceedings of the thirteenth international conference on artificial intelligence and statistics**, 2010, pp. 249–256.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification", in **Proceedings of the IEEE international conference on computer vision**, 2015, pp. 1026–1034.

[43] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?", **Journal of Machine Learning Research**, vol. 11, no. Feb, pp. 625–660, 2010.

[44] N. Qian, "On the momentum term in gradient descent learning algorithms", **Neural networks**, vol. 12, no. 1, pp. 145–151, 1999.

[45] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o (1/k2̂)", in **Doklady AN USSR**, vol. 269, 1983, pp. 543–547.

[46] R. Kuditipudi, X. Wang, H. Lee, Y. Zhang, Z. Li, W. Hu, S. Arora, and R. Ge, "Explaining landscape connectivity of low-cost solutions for multilayer nets", **arXiv preprint arXiv:1906.06247**, 2019.

[47] J. Cavazza, P. Morerio, B. Haeffele, C. Lane, V. Murino, and R. Vidal, "Dropout as a low-rank regularizer for matrix factorization", **arXiv preprint arXiv:1710.05092**, 2017.

[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", **The journal of machine learning research**, vol. 15, no. 1, pp. 1929–1958, 2014.