Memorization in Neural Networks: Does the Loss Function Matter?

Deep Patel, P.S. Sastry

Learning Systems and Multimedia Lab
Department of Electrical Engineering
Indian Institute of Science, Bangalore - 560 012

PAKDD 2021

‘Memorization’ in Deep networks

- Deep neural networks are very good at interpolating the data

‘Memorization’ in Deep networks

- Deep neural networks are very good at interpolating the data
- Zhang et al. [11] showed that SGD-based training of neural networks drives the training accuracy to 100% even in case of randomly labelled data.

‘Memorization’ in Deep networks

- Deep neural networks are very good at interpolating the data
- Zhang et al. [11] showed that SGD-based training of neural networks drives the training accuracy to 100% even in case of randomly labelled data.
- They call this ‘memorization’ as the network overfits to the training data. (However, there is no precise definition in the paper).

‘Memorization’ in Deep networks

- Deep neural networks are very good at interpolating the data
- Zhang et al. [11] showed that SGD-based training of neural networks drives the training accuracy to 100% even in case of randomly labelled data.
- They call this ‘memorization’ as the network overfits to the training data. (However, there is no precise definition in the paper).
- None of the standard regularization methods such as weight decay, drop-out, etc. seem effective in resisting such overfitting.

'Memorization' in Deep networks

- Deep neural networks are very good at interpolating the data
- Zhang et al. [11] showed that SGD-based training of neural networks drives the training accuracy to 100% even in case of randomly labelled data.
- They call this 'memorization' as the network overfits to the training data. (However, there is no precise definition in the paper).
- None of the standard regularization methods such as weight decay, drop-out, etc. seem effective in resisting such overfitting.
- Many other studies (e.g., [1, 5, 3, 4]) throw interesting light on the dynamics of this memorization process and what it means for generalization in deep networks

Our Study

- Here, we study the effect of loss function on degree of memorization

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?
- Depends on the kind of local minima that SGD process can take the network to

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?
- Depends on the kind of local minima that SGD process can take the network to
- Depends on the topography of empirical risk that is minimized.

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?
- Depends on the kind of local minima that SGD process can take the network to
- Depends on the topography of empirical risk that is minimized.
- The choice of loss function can be critical in determining this.

Our Study

- Here, we study the effect of loss function on degree of memorization
- Memorization is essentially about training error.
- It is about: Can the network (always) learn a function that perfectly interpolates data?
- Depends on the kind of local minima that SGD process can take the network to
- Depends on the topography of empirical risk that is minimized.
- The choice of loss function can be critical in determining this.
- None of the studies on memorization investigate this.

Our Results

- Choice of loss function can affect memorization

Our Results

- Choice of loss function can affect memorization
- We show empirically that a symmetric loss function can resist memorization to a good degree

Our Results

- Choice of loss function can affect memorization
- We show empirically that a symmetric loss function can resist memorization to a good degree
- We formally define what 'resisting memorization' means and provide some theoretical justification for the empirical results

Some Comments

- There are many studies on the relative efficacy of different loss functions for classification and regression (e.g., [6, 2, 9, 10])

Some Comments

- There are many studies on the relative efficacy of different loss functions for classification and regression (e.g., [6, 2, 9, 10])
- Our study is distinct – what role loss function can play in affecting the degree of memorization in overparameterized networks?

Some Comments

- There are many studies on the relative efficacy of different loss functions for classification and regression (e.g., [6, 2, 9, 10])
- Our study is distinct – what role loss function can play in affecting the degree of memorization in overparameterized networks?
- Design of algorithms for robust learning when training data has randomly corrupted labels, is also a much studied problem

Some Comments

- There are many studies on the relative efficacy of different loss functions for classification and regression (e.g., [6, 2, 9, 10])
- Our study is distinct – what role loss function can play in affecting the degree of memorization in overparameterized networks?
- Design of algorithms for robust learning when training data has randomly corrupted labels, is also a much studied problem
- Here our interest is in the inherent ability of a loss function to resist overfitting of training data when labels are randomly altered.

Notation

- $\mathcal{X} \subseteq \mathbb{R}^n$: feature space;
- $\mathcal{Y} = \{1, \ldots, K\}$ where K: number of classes
- $S = \{\boldsymbol{x}_i, y_i^{cl}\}_{i=1}^{\ell}$: Original training set
- $S_\eta = \{\boldsymbol{x}_i, y_i\}_{i=1}^{\ell}$: Training set with randomly altered labels:

$$y_i = \begin{cases} y_i^{cl} & \text{with probability } 1 - \eta \\ j \in \mathcal{Y} - \{y_i^{cl}\} & \text{with probability } \frac{\eta}{K-1} \end{cases} \quad (1)$$

  where η is referred to as the noise rate.
- h_η: classifier function (with softmax layer as output layer) learnt by an algorithm with S_η as training data
- To define loss functions we take y_i to be one-hot vector. ($\boldsymbol{e}^k$ represents class-k).

Notation (contd.)

- Let $J_1 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\boldsymbol{x}_i)=y_i]}$
- This is the usual training accuracy of classifier h_η on the training set with randomly altered labels.

Notation (contd.)

- Let $J_1 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\mathbf{x}_i) = y_i]}$
- This is the usual training accuracy of classifier h_η on the training set with randomly altered labels.
- Let $J_2 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\mathbf{x}_i) = y_i^{cl}]}$
- This is the accuracy of h_η on the same training set but computed with respect to the original labels.

Notation (contd.)

- Let $J_1 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\boldsymbol{x}_i) = y_i]}$
- This is the usual training accuracy of classifier h_η on the training set with randomly altered labels.
- Let $J_2 = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbb{I}_{[h_\eta(\boldsymbol{x}_i) = y_i^{cl}]}$
- This is the accuracy of h_η on the same training set but computed with respect to the original labels.
- Relative values of J_1 and J_2 can give interesting insights into how different loss functions behave.

Loss Functions We Compare

- We consider 3 loss functions in this paper: CCE, MSE, and Robust Log Loss (RLL).

$$\begin{aligned}
\mathcal{L}_{CCE}(\mathbf{h}(\mathbf{x}), \mathbf{e}^k) &= -\sum_i e_i^k \, \log\left(h_i(\mathbf{x})\right) = -\log(h_k(\mathbf{x})) \\
\mathcal{L}_{MSE}(\mathbf{h}(\mathbf{x}), \mathbf{e}^k) &= \sum_i \left(h_i(\mathbf{x}) - e_i^k\right)^2 \\
\mathcal{L}_{RLL}(\mathbf{h}(\mathbf{x}), \mathbf{e}^k) &= \log\left(\frac{\alpha+1}{\alpha}\right) - \log(\alpha + h_k(\mathbf{x})) \\
&\quad + \sum_{j \neq k} \frac{1}{K-1} \log(\alpha + h_j(\mathbf{x}))
\end{aligned}$$

where $\alpha > 0$ is a parameter of the RLL.

Loss Functions (contd.)

- CCE and MSE are fairly commonly used loss functions for classification and regression tasks.
- RLL is a **symmetric** loss.

Symmetric Loss

A loss function, L, is **symmetric** if $\exists\ C \in \mathbb{R}_{++}$ such that:

$$\sum_{j=1}^{K} L(h(\boldsymbol{x}), j) = C,\ \forall h, \boldsymbol{x} \tag{2}$$

We next present some empirical results comparing these three losses in their ability to resist memorization

Experimental Setup

Datasets:

- MNIST [8]
- CIFAR-10 [7]

Networks:

- Inception-Lite (same as that used in [11]) for CIFAR-10
- ResNet-32 for CIFAR-10
- ResNet-18 for MNIST

Experimental Setup (contd.)

Details about network training:

Inception-Lite

- SGD(lr $= 10^{-2}$, momentum=0.9)
- number of epochs = 100
- learning rate reduced by 0.95 factor every epoch

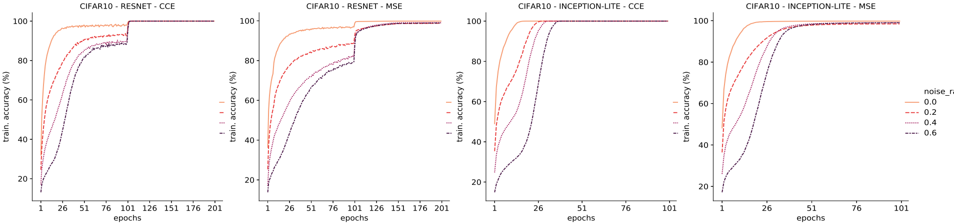
ResNet-32

- SGD(lr $= 10^{-1}$, momentum=0.9, weight_decay=10^{-4})
- number of epochs = 200
- learning rate reduced by 0.1 factor at epochs 100 and 150

ResNet-18

- Adam(lr $= 10^{-3}$)
- number of epochs = 200

Results on CIFAR-10

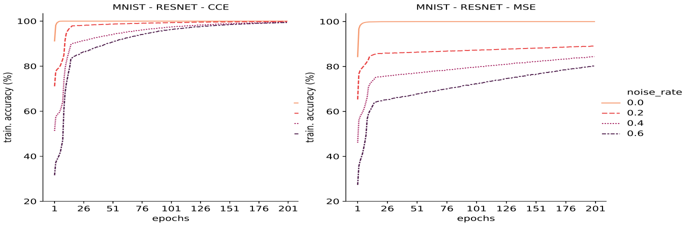


(a) *CCE* **(b)** *MSE* **(c)** *CCE* **(d)** *MSE*

Figure 1: Training set accuracies for ResNet-32 ((a) & (b)) & Inception-Lite ((c) & (d)) trained on CIFAR-10 with CCE and MSE losses for for $\eta \in \{0., 0.2, 0.4, 0.6\}$

- CCE and MSE achieve 100% training accuracy (irrespective of noise rate) thus showing they memorize random labels.

Results on MNIST

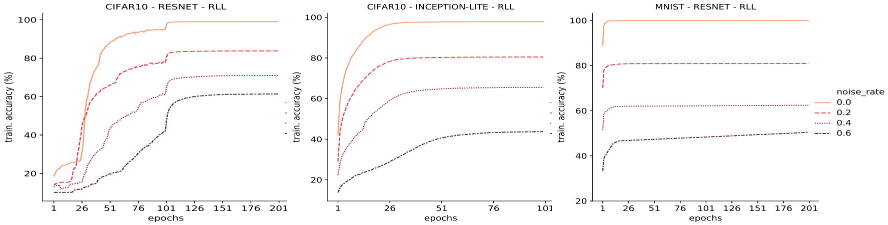


(a) *CCE* **(b)** *MSE*

Figure 2: Training set accuracies for ResNet-18 trained on MNIST with CCE and MSE losses for $\eta \in \{0., 0.2, 0.4, 0.6\}$

- CCE overfits even with this smaller network. MSE also achieves high trainig accuracy irrespective of amount of noise.

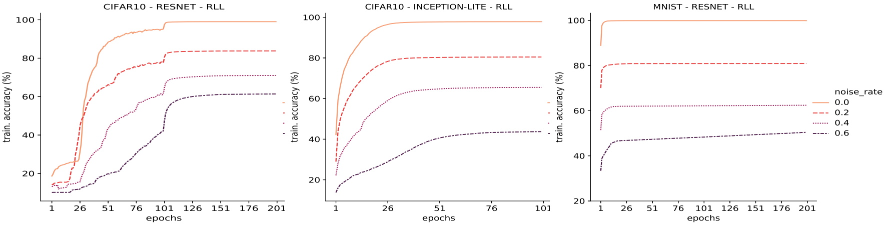
Results on CIFAR-10 & MNIST - with RLL



(a) $ResNet - 32$ **(b)** $Inception - Lite$ **(c)** $ResNet - 18$

Figure 3: Training set accuracies for networks trained on CIFAR-10 ((a) & (b)) and MNIST ((a)) with RLL for $\eta \in \{0., 0.2, 0.4, 0.6\}$

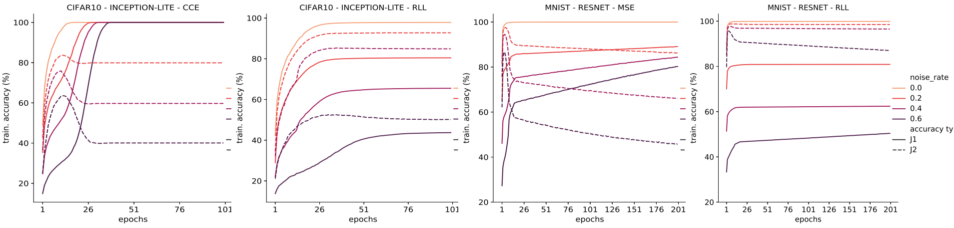
Results on CIFAR-10 & MNIST - with RLL



(a) $ResNet-32$ **(b)** $Inception-Lite$ **(c)** $ResNet-18$

- Training accuracies of RLL saturate much below 100% – the higher the noise rate the lower the training accuracy
- We can see from Figure 4 that the training accuracy of RLL saturates to almost $(1-\eta) \times 100\%$ (η :– noise rate). Now the network does not overfit. In addition, it is as if we have almost inferred the noise rate!

Results for J_1 and J_2 accuracy - with RLL



(a) *CCE* **(b)** *RLL* **(c)** *MSE* **(d)** *RLL*

Figure 5: J_1 and J_2 accuracies for $\eta \in \{0., 0.2, 0.4, 0.6\}$ (Solid lines show J_1 accuracy; dashed lines show J_2 accuracy)

- For RLL, the J_2 is always above J_1 curve – showing RLL resists overfitting to noisy labels
- For CCE and MSE, the J_1 curve eventually goes above J_2 – showing the network overfits the noisy labels as epochs progress

Resisting Memorization

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.

Resisting Memorization

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.
- We say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h(\boldsymbol{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h_\eta(\boldsymbol{x}_i)=y_i^{cl}\}}$$

Resisting Memorization

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.
- We say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h(\boldsymbol{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h_\eta(\boldsymbol{x}_i)=y_i^{cl}\}}$$

- This is a reasonable formalization for 'resisting memorization'

Resisting Memorization

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.
- We say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h(\boldsymbol{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h_\eta(\boldsymbol{x}_i)=y_i^{cl}\}}$$

- This is a reasonable formalization for ‘resisting memorization’
- Note that the J_2 accuracy defined earlier is the quantity on RHS above. The accuracy for $\eta = 0$ would be the quantity on LHS above.

Resisting Memorization

- Recall: S is the original training data and S_η is the data with randomly altered labels; $\mathcal{D}$ and $\mathcal{D}_\eta$ are the corresponding distributions.
- Let h and h_η denote the classifier learned by an algorithm when given S and S_η as training data, respectively.
- We say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{I}_{\{h(\mathbf{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}_{\{h_\eta(\mathbf{x}_i)=y_i^{cl}\}}$$

- This is a reasonable formalization for 'resisting memorization'
- Note that the J_2 accuracy defined earlier is the quantity on RHS above. The accuracy for $\eta = 0$ would be the quantity on LHS above.
- For RLL, as we saw, the J_2 accuracy is mostly close to the accuracy achieved when $\eta = 0$

Resisting Memorization (contd.)

Theorem

Let $\mathcal{L}$ be a symmetric loss, $\mathcal{D}$ and $\mathcal{D}_\eta$ as defined above. Assume $\eta < \frac{K-1}{K}$. Let $y_{\boldsymbol{x}}^{cl}$ and $y_{\boldsymbol{x}}$ denote the original and noisy labels corresponding to the pattern $\boldsymbol{x}$. The risk of h over $\mathcal{D}$ and $\mathcal{D}_\eta$ is $R_{\mathcal{L}}(h) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}}^{cl})]$ and $R_{\mathcal{L}}^{n}(h) = \mathbb{E}_{\mathcal{D}_\eta}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}})]$ resp. Then, given any two classifiers h_1 and h_2, if $R_{\mathcal{L}}(h_1) < R_{\mathcal{L}}(h_2)$, then $R_{\mathcal{L}}^{\eta}(h_1) < R_{\mathcal{L}}^{\eta}(h_2)$ and vice versa.

Resisting Memorization (contd.)

Theorem

Let $\mathcal{L}$ be a symmetric loss, $\mathcal{D}$ and $\mathcal{D}_\eta$ as defined above. Assume $\eta < \frac{K-1}{K}$. Let $y_{\boldsymbol{x}}^{cl}$ and $y_{\boldsymbol{x}}$ denote the original and noisy labels corresponding to the pattern $\boldsymbol{x}$. The risk of h over $\mathcal{D}$ and $\mathcal{D}_\eta$ is $R_{\mathcal{L}}(h) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}}^{cl})]$ and $R_{\mathcal{L}}^{n}(h) = \mathbb{E}_{\mathcal{D}_\eta}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}})]$ resp. Then, given any two classifiers h_1 and h_2, if $R_{\mathcal{L}}(h_1) < R_{\mathcal{L}}(h_2)$, then $R_{\mathcal{L}}^{\eta}(h_1) < R_{\mathcal{L}}^{\eta}(h_2)$ and vice versa.

- For symmetric losses relative risks of two classifiers are same both with and without noise.

Resisting Memorization (contd.)

Theorem

Let $\mathcal{L}$ be a symmetric loss, $\mathcal{D}$ and $\mathcal{D}_\eta$ as defined above. Assume $\eta < \frac{K-1}{K}$. Let $y_{\boldsymbol{x}}^{cl}$ and $y_{\boldsymbol{x}}$ denote the original and noisy labels corresponding to the pattern $\boldsymbol{x}$. The risk of h over $\mathcal{D}$ and $\mathcal{D}_\eta$ is $R_{\mathcal{L}}(h) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}}^{cl})]$ and $R_{\mathcal{L}}^{n}(h) = \mathbb{E}_{\mathcal{D}_\eta}[\mathcal{L}(h(\boldsymbol{x}), y_{\boldsymbol{x}})]$ resp. Then, given any two classifiers h_1 and h_2, if $R_{\mathcal{L}}(h_1) < R_{\mathcal{L}}(h_2)$, then $R_{\mathcal{L}}^{\eta}(h_1) < R_{\mathcal{L}}^{\eta}(h_2)$ and vice versa.

- For symmetric losses relative risks of two classifiers are same both with and without noise.
- So, symmetric losses can resist memorization (if we can get minimum of risk)

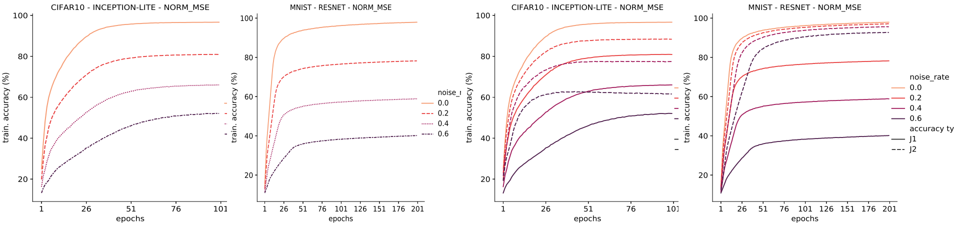
Resisting Memorization: Symmetric Losses

- As is easy to see, the symmetry condition implies that the loss function is bounded.
- Given a bounded loss function we can satisfy the symmetry condition by 'normalizing' it. Given a bounded loss, L, define $\bar{L}$, by

$$\bar{L}(h(X), j) = \frac{L(h(X), j)}{\sum_s L(h(X), s)} \tag{3}$$

- $\bar{L}$ satisfies the symmetry condition (Equation 2). As mentioned earlier, CCE loss is unbounded and hence normalization would not turn it into a symmetric loss. However, we can normalize MSE loss.

Results for J_1 and J_2 accuracy - with Norm. MSE



(a) *Norm.MSE* **(b)** *Norm.MSE* **(c)** *Norm.MSE* **(d)** *Norm.MSE*

Figure 6: Train. accuracy and J_1 & J_2 accuracies for Inception-Lite ((a) & (c)) & ResNet-18 ((b) & (d)) trained on CIFAR-10 and MNIST resp. for $\eta \in \{0., 0.2, 0.4, 0.6\}$ (Solid lines show J_1 accuracy; dashed lines show J_2 accuracy)

- Once we normalize MSE, it no longer overfits the data with random labels; it behaves more like RLL now.

Conclusions

- The phenomenon of memorization in deep networks has received a lot of attention because it raises important questions on how to understand generalization abilities of these networks.
- In this work we have shown through empirical studies that changing the loss function alone can significantly change this memorization.
- We showed this with the symmetric loss function, RLL, and we have provided some theoretical analysis to explain the empirical results.
- The results presented here suggest that choice of loss function can play a critical role in overfitting by deep networks.
- We feel it is important to further investigate the nature of different (symmetric) loss functions for a better understanding of robust learning.

Thank You
Any Questions?

References I

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al.
A closer look at memorization in deep networks.
In *International Conference on Machine Learning*, pages 233–242. PMLR, 2017.

A. Demirkaya, J. Chen, and S. Oymak.
Exploring the role of loss functions in multiclass classification.
In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5, 2020.

Vitaly Feldman.
Does learning require memorization? a short tale about a long tail.
In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

References II

Vitaly Feldman and Chiyuan Zhang.
What neural networks memorize and why: Discovering the long tail via influence estimation.
Advances in Neural Information Processing Systems, 33, 2020.

Jindong Gu and Volker Tresp.
Neural network memorization dissection, 2019.

Like Hui and Mikhail Belkin.
Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks, 2020.

Alex Krizhevsky.
Learning Multiple Layers of Features from Tiny Images.
PhD thesis, University of Toronto, 2009.

References III

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.

Qichao Que and Mikhail Belkin.
Back to the future: Radial basis function networks revisited.
In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1375–1383, Cadiz, Spain, 09–11 May 2016. PMLR.

Ryan Michael Rifkin.
Everything old is new again: a fresh look at historical approaches in machine learning.
PhD thesis, Massachussets Insitute of Technology, 2002.

References IV

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals.
Understanding deep learning requires rethinking generalization.
arXiv preprint arXiv:1611.03530, 2016.