OPT2021: 13th Annual Workshop on Optimization for Machine Learning 1–10

Understanding Memorization from the Perspective of Optimization via Efficient Influence Estimation

Futong Liu FUTONG.LIU@EPFL.CH
Tao Lin* TAO.LIN@EPFL.CH
Martin Jaggi MARTIN.JAGGI@EPFL.CH
EPFL, Switzerland

Abstract

Over-parameterized deep neural networks are able to achieve excellent training accuracy while maintaining a small generalization error. It has also been found that they are able to fit arbitrary labels, and this behaviour is referred to as the phenomenon of memorization. In this work, we study the phenomenon of memorization with turn-over dropout, an efficient method to estimate influence and memorization, for data with true labels (real data) and data with random labels (random data). Our main findings are: (i) For both real data and random data, the optimization of easy examples (e.g., real data) and difficult examples (e.g., random data) are conducted by the network simultaneously, with easy ones at a higher speed; (ii) For real data, a correct difficult example in the training dataset is more informative than an easy one. By showing the existence of memorization on random data and real data, we highlight the consistency between them regarding optimization and we emphasize the implication of memorization during optimization.

1. Introduction

Over-parameterized deep neural networks (DNNs), which have more number of trainable parameters than that of data instances, usually achieve excellent performance on both the training dataset and the test dataset. However, it has also been found that a DNN has the capability to perfectly fit pure random labels, and this behaviour of a DNN when trained on random labels or random inputs has been implicitly referred to as the phenomenon of memorization (Zhang et al., 2017; Arpit et al., 2017; Brown et al., 2021; Maennel et al., 2020). Understanding the memorization of deep neural networks is a key step towards characterizing its optimization dynamics and interpreting the model's generalization ability.

Zhang et al. (2017) formulated the memorization phenomenon by showing that a neural network with sufficient capacity can memorize completely random labels or random noise data (inputs) without substantially longer training time, and the explicit regularization is neither necessary nor sufficient to control the generalization error. Arpit et al. (2017) further varied the amount of random labels, and concluded that DNNs first learn general patterns of the real data before fitting the random data.

The phenomenon of memorization can also be observed on a real dataset when there are no random labels. Feldman and Zhang (2020) studied memorization through the lens of self-influence estimation, and provided compelling evidences for the long tail theory (Feldman, 2020). They believe that memorization is necessary for achieving close-to-optimal general-

* Corresponding author

ization error when the data distribution is long-tailed. Modern datasets usually follows a long tail distribution, where the subpopulations at the head of the distribution include a large amount of frequent and easy examples while the subpopulations at the tail include rare and difficult examples.

However, the insights of these prior work normally are limited through some computationally intensive estimation. For example, Arpit et al. (2017); Gu and Tresp (2019) only study the memorization phenomenon on random data, and it is not clear if such observations can be generalized to real data. Similarly, the study on influence estimation (Feldman and Zhang, 2020; Koh and Liang, 2017) focuses on the cross-influence of a training instance over a test instance, instead of self-influence nor its implication on optimization. To this end, we introduce the memorization score (i.e. self-influence)—as a unified and computationally efficient metric—on both real data and random data to estimate the difficulty of training instances; such efficient estimation allows us to study the interpolation between memorization, optimization, and its implication on generalization.

Our main contributions are summarized as follows:

1. We leverage turn-over dropout to quantify the degree of memorization by estimating its memorization score. We show it is feasible and efficient on different network architectures.
2. For real data and random data, easy and difficult examples are learnt simultaneously. However, the optimization on easy examples converges faster than that of difficult ones, due to the higher gradient similarity between easy examples.
3. For real data, difficult examples contain information about easy ones but easy examples have very little information about difficult ones. Easy examples can benefit from both easy and difficult examples, and hence the optimization speed of easy examples is faster.

2. Influence Estimation with Turn-over Dropout

2.1. Preliminary

We denote $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ as the training dataset, $\mathcal{X}$ as the input space (e.g., images) and $\mathcal{Y}$ as output space (e.g., labels). Let $\mathbf{z}_i := (\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{z}$ is a data example and represent a pair of input $\mathbf{x}_i \in \mathcal{X}$ and its output $\mathbf{y}_i \in \mathcal{Y}$. A model trained on dataset $\mathcal{D}$ is denoted as $f_\mathcal{D} : \mathcal{X} \to \mathcal{Y}$. The loss function $\mathcal{L}(f, \mathbf{z}_i)$ represents the loss of the model's prediction on $\mathbf{x}_i$ over its label $\mathbf{y}_i$.

The phenomenon of memorization and generalization can be quantified by the **leave-one-out influence** function (Feldman and Zhang, 2020; Koh and Liang, 2017), which measures the influence of example $\mathbf{z}_i$ to the prediction of $\mathbf{z}_{target}$. Let $f_\mathcal{D}$ be the model trained on the full training dataset $\mathcal{D}$ and $f_{\mathcal{D}\setminus\{\mathbf{z}_i\}}$ be the model trained on $\mathcal{D}$ excluding example $\mathbf{z}_i$, then the influence is defined as:

$$\mathcal{I}(\mathbf{z}_{target}, \mathbf{z}_i; \mathcal{D}) := \mathcal{L}(f_{\mathcal{D}\setminus\{\mathbf{z}_i\}}, \mathbf{z}_{target}) - \mathcal{L}(f_\mathcal{D}, \mathbf{z}_{target}) \,, \tag{1}$$

where $\mathbf{z}_{target}$ can be an instance in the training set or the test set. When $\mathbf{z}_{target} = \mathbf{z}_i \in \mathcal{D}$, **memorization** can be defined as **self-influence** (Feldman and Zhang, 2020):

$$\mathcal{I}(\mathbf{z}_i, \mathbf{z}_i; \mathcal{D}) := \mathcal{L}(f_{\mathcal{D}\setminus\{\mathbf{z}_i\}}, \mathbf{z}_i) - \mathcal{L}(f_\mathcal{D}, \mathbf{z}_i) \,. \tag{2}$$

The memorization score computed by Equation (2) indicates the extent of memorization. A large memorization score means that $\mathbf{z}_i$ is memorized to be fitted, as the model that is

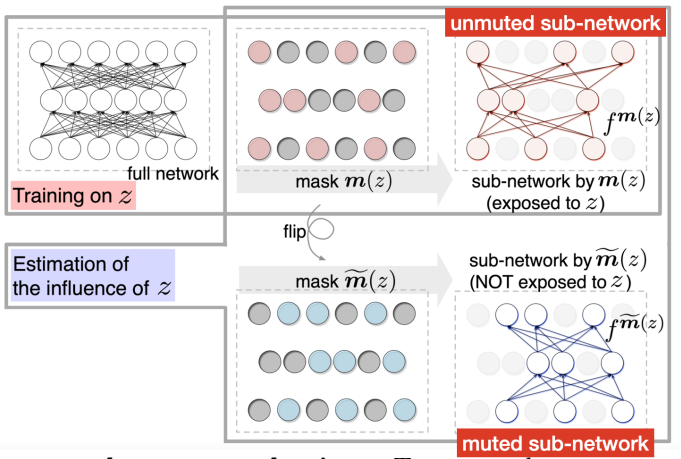


Figure 1: **The turn-over dropout mechanism**. Turn-over dropout generates a deterministic mask for each training example z and only updates the un-muted sub-network. The muted sub-network is not updated with respect to z. The influence estimation is the loss difference of the two sub-networks. Cited and adapted from Kobayashi et al. (2020).

trained without $\mathbf{z}_i$ cannot correctly classify $\mathbf{z}_i$. A small score indicates that the label of $\mathbf{z}_i$ can be inferred from other training data, and hence $\mathbf{z}_i$ is part of a generalizable pattern learnt by the model.

For a more fine-grained understanding, we define the **cosine similarity** of a set of gradient vectors $\{v_i\}, i \in [n]$ as the average pair-wise cosine similarities, as

$$\text{cosine similarity} = \mathbb{E}_{i,j}\left[\frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\|_2 \cdot \|\mathbf{v}_j\|_2}\right], \quad \forall i, j \in [n] \; and \; i \neq j \,. \tag{3}$$

We further define the **contribution** of an example's gradient to the actual gradient update step as the projection of the example's gradient on the gradient of the mini-batch $\mathbf{g}$, i.e.

$$\text{contribution of } \mathbf{v}_i = (\mathbf{v}_i \cdot \mathbf{g}) / \|\mathbf{g}\|_2 \,. \tag{4}$$

2.2. Turn-over Dropout as an Efficient Estimation for Memorization

It is computationally infeasible to calculate the exact leave-one-out influence and memorization scores through Equation (1) and (2), due to the significant overhead introduced by training the extra model from scratch for each training instance.

For the ease of getting a deeper understanding on the optimization and generalization (as our main contribution), we follow the idea of **turn-over dropout** (Kobayashi et al., 2020), which is stated as follows (and the workflow refers to Figure 1). We also visualize some cases of easy and difficult examples identified by turn-over dropout in Appendix C.

The turn-over dropout generates a *deterministic* mask $\mathbf{m}(\mathbf{z}_i)$ for each training instance $\mathbf{z}_i$, and the mask is fixed throughout training and evaluation. The masks are controlled to ensure no two training instances will share the same mask, and each mask splits the model into two sub-networks: the muted sub-network and the unmuted sub-network.

We use $f^{\tilde{\mathbf{m}}(\mathbf{z}_i)}$ and $f^{\mathbf{m}(\mathbf{z}_i)}$ to approximate $f_{\mathcal{D}\setminus\{\mathbf{z}_i\}}$ and $f_{\mathcal{D}}$ respectively, given the fact that the muted sub-network $f^{\tilde{\mathbf{m}}(\mathbf{z}_i)}$ is updated by $\mathcal{D}\setminus\{\mathbf{z}_i\}$ and the un-muted sub-network $f^{\mathbf{m}(\mathbf{z}_i)}$ is updated by $\mathcal{D}$. Hence, the influence score in Equation (1) can be estimated by

$$\mathcal{I}(\mathbf{z}_{target}, \mathbf{z}_i; \mathcal{D}) \approx \mathcal{L}(f^{\tilde{\mathbf{m}}(\mathbf{z}_i)}, \mathbf{z}_{target}) - \mathcal{L}(f^{\mathbf{m}(\mathbf{z}_i)}, \mathbf{z}_{target}) \,, \tag{5}$$

and the memorization (self-influence) score can be estimated by

$$\mathcal{I}(\mathbf{z}_i, \mathbf{z}_i; \mathcal{D}) \approx \mathcal{L}(f^{\tilde{\mathbf{m}}(\mathbf{z}_i)}, \mathbf{z}_i) - \mathcal{L}(f^{\mathbf{m}(\mathbf{z}_i)}, \mathbf{z}_i)\,. \tag{6}$$

Note that the prior work (Kobayashi et al., 2020) uses turn-over dropout as an addition to DNNs to increase the model's interpretability. With turn-over dropout only added to deep layers, their implementation does not sacrifice the model's performance too much. Our work instead applies turn-over dropout to all layers and uses the adapted model as a proxy to estimate self-influence. Furthermore, we extend the self-influence information to the training procedure and study its implication on optimization.

3. Observations and Insights

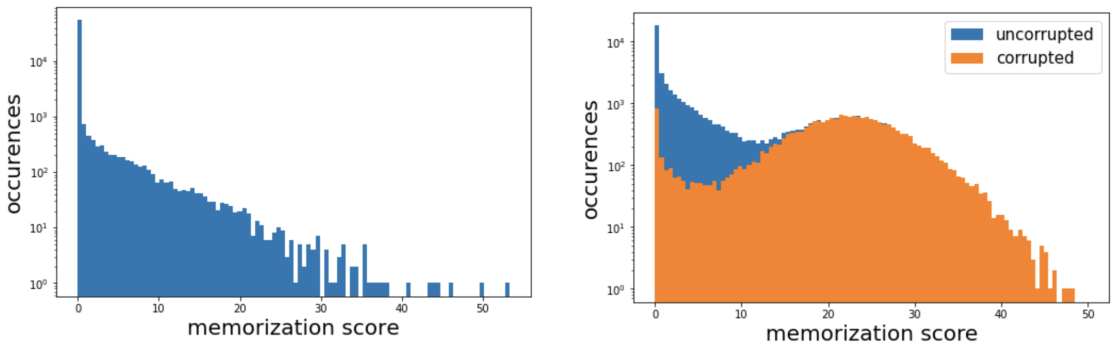
3.1. Unify random data and real data by memorization scores

A commonly-used method to study memorization is to *corrupt* the training data by replacing part of true labels with random labels. With this method, previous work (Zhang et al., 2017; Arpit et al., 2017; Maennel et al., 2020) focuses on comparing the *differences* between the scenario of label corruption and the scenario of no corruption to understand memorization. On the contrary, we juxtapose the two scenarios and focus on comparing the easy and difficult examples to find what is in *common* regarding optimization.

A corrupted dataset is a more difficult version of the uncorrupted one. Figure 2 shows the histogram of the memorization scores (computed from turn-over dropout) on uncorrupted MNIST and corrupted MNIST. When there is no corruption, the scores follow a long tail distribution as described in Feldman (2020) (c.f. Figure 2(a)). Corrupting an instance significantly increases its difficulty (memorization score): due to the random pattern/relationship between the image and the label, the model has no option but to *memorize* the instance. Random examples in the corrupted scenario are the counterpart of the difficult examples in the uncorrupted scenario, as shown in Figure 2(b). Hence, *random data and real data are consistent with respect to memorization scores*, and the methodology to study the properties of random data sheds light on difficult examples in the uncorrupted scenario.

3.2. Training on random labels

In this section, our investigation starts from the revisiting on the experiments of Arpit et al. (2017) by corrupting the dataset with different degree of corruption.



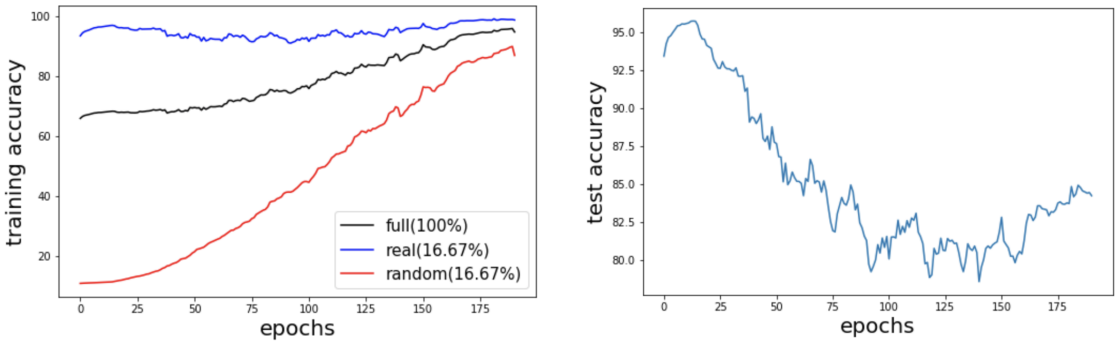
(a) Uncorrupted MNIST. (b) MNIST corrupted with 20,000 random labels.

Figure 2: **Memorization scores of MNIST** computed from a one-hidden layer MLP with turn-over dropout. (a) Data follows a long tail distribution. (b) Data with random labels (corrupted) have higher memorization scores than data with real labels (uncorrupted).

Easy examples are learnt faster. In contrast to the common belief introduced in Arpit et al. (2017)[1] that a model first learns the simple and general patterns (easy examples) before fitting the noise (difficult examples), we provide a refined insight below: *easy and difficult examples are learnt by the model simultaneously, with easy examples learnt faster.* As indicated in Section 3.1, in the presence of label corruption, data with random labels are of the highest difficulty.

Figure 3 shows the training and test dynamics of a one-hidden-layer MLP on the MNIST dataset with 20,000 instances replaced by random labels. We keep track of training dynamics of an easy subset and a difficult subset of the training data. It can be observed that easy examples are learnt much faster than difficult ones. The build-up of the speed difference reflects on the dynamic of test accuracy: the test accuracy reaches the peak (at around 20th epoch), and that is when the model has successfully fitted the real data but barely fitted the random noise yet.

However, we also observe from Figure 3(a) (and Figure 5 later) that there are small improvement on the difficult subset at the initial epochs. Albeit the improvement is small, it indicates that difficult examples are in fact learnt simultaneously with easy examples, and Figure 4(b) in the next section shows stronger proof on this point.



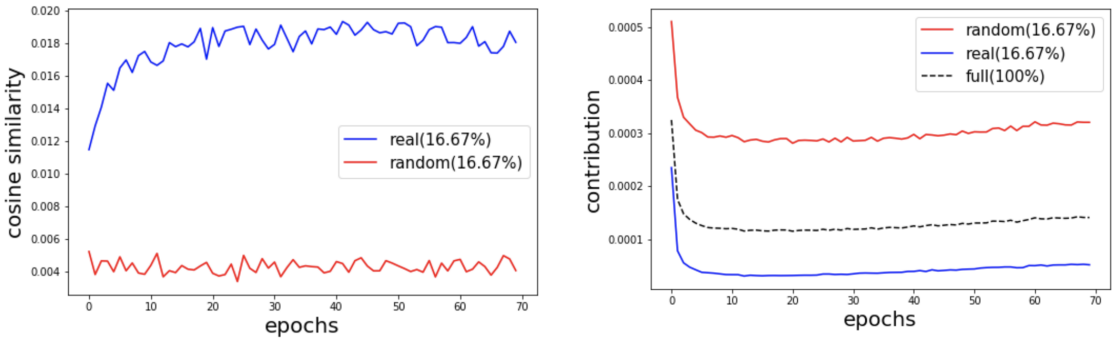
(a) Training accuracy. (b) Test accuracy.

Figure 3: **The training and test dynamics** of a one-hidden-layer **MLP on the MNIST** dataset when 20,000 of the training dataset is corrupted by random labels. Results are sliding-window smoothed. The black line ("full") in (a) and the line in (b) shows the dynamics on the entire training/test dataset respectively. (a) also keeps track of the training accuracy of a 10,000 (16.67%) real-data subset (blue) and a 10,000 (16.67%) random-label subset (red). The percentage in the legend indicates the size of the subset. (b) shows that the test accuracy reaches the peak with the build-up of the speed difference and drops with the random data improving its training accuracy.

Why are easy examples learnt faster than difficult examples? Below we try to answer the question from the perspective of optimization (SGD updates) of real data and random data[2]. Figure 4 shows the evolution of the average cosine similarity (Equation (3)) and the average contribution (Equation (4)) of easy examples and difficult examples.

The cosine similarity between easy examples is much higher than that of difficult ones, as shown in Figure 4(a). Easy examples reside at the head of the long tail distribution and belong to similar subpopulations, and this similarity between images engenders similar

1. We also reproduce these experiments in Appendix E.
2. We use Backpack (Dangel et al., 2020) to extract the gradient of each example in the batch during the backward pass.



(a) Cosine similarity (b) Contribution

Figure 4: Dynamics of **cosine similarity** and **contribution** of an easy (real) subset and a difficult (random) subset of the training data (**MLP** trained on **MNIST** with 20,000 instances corrupted). (a) The cosine similarity between easy examples is much higher. (b) The difficult examples are the driving force of SGD update throughout training.

gradients. These similar gradients strengthen each other to make a SGD update that can benefit them all, which results in the faster optimization speed for easy examples. This observation is in agreement with Chatterjee (2020), who uses a measure of gradient coherence to show that the parameter update of the network can benefit many examples when such similarity exists.

Additionally, we observe from Figure 4(b) that the average contribution of difficult examples is constantly higher than easy examples and the entire dataset. The high contribution of difficult examples throughout training shows that difficult examples are learnt simultaneously with easy examples, in contrast to the prior belief that DNNs prioritize the learning of easy examples.

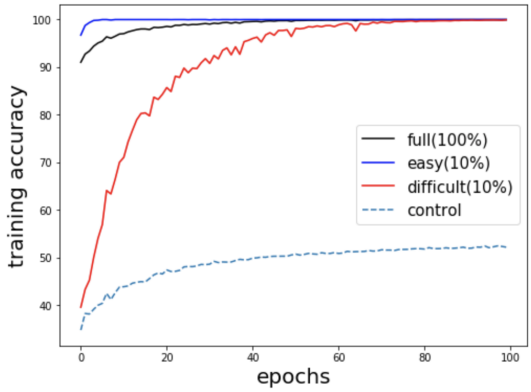
3.3. Training on real data

Training dynamics on real data. Figure 5 shows how the easy/difficult/full subsets of the training/test data evolve with respect to epochs on MNIST and CIFAR-10. We compute the memorization scores of the training data and test data beforehand, and monitor an easy subset and a difficult subset during training and validation. The observation is consistent with the random-label case: *easy examples and difficult examples are learnt simultaneously, with easy examples fitted faster than difficult ones.*

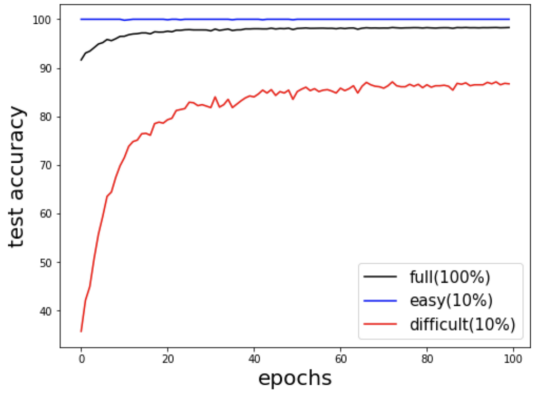
Figure 5(b) and 5(d) shows the test dynamics are similar to the training dynamics: *easy examples benefit from learning quicker than difficult ones and reach a higher test accuracy.* This similarity reflects the consistency of the subpopulations identified.

Difficult examples are more informative. As explained in the random label setting (Section 3.2), easy examples are learnt faster than difficult ones due to the higher gradient similarity of easy examples. Complementing this observation, we further highlight a new insight in Figure 6 below for real data, which again explains *the faster convergence on the easy examples*, that is *difficult examples contain information about the easy ones while the easy examples have very little information about the difficult ones.*

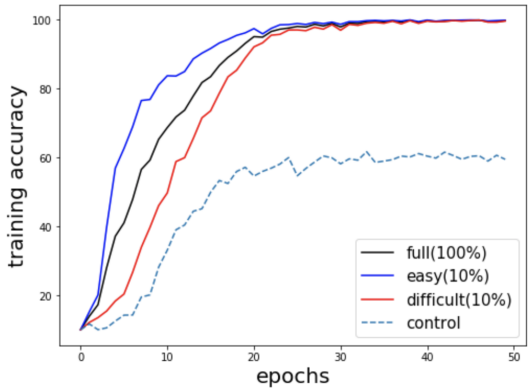
Figure 6 presents the training dynamics of two manually-constructed sub-datasets: with the memorization scores computed for the training data, we sample an easy subset and a difficult subset that are of the same size from the whole training dataset; and for each time, we train with one subset, and use the other one as the validation set. We can witness from



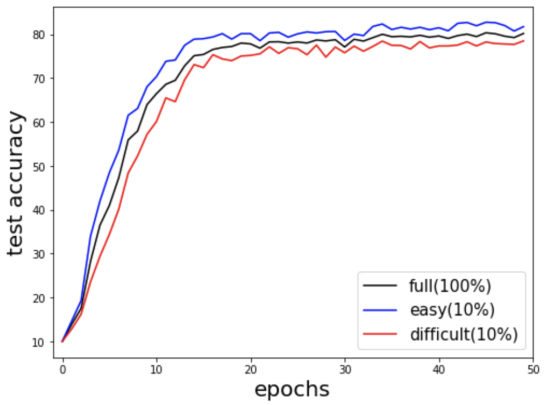
(*a*) Training accuracy on MNIST.



(*b*) Test accuracy on MNIST.



(*c*) Training accuracy on CIFAR-10.

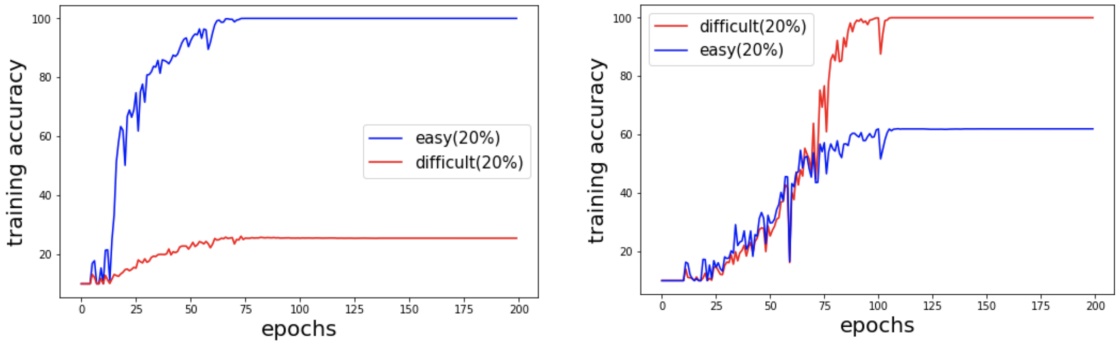


(*d*) Test accuracy, CIFAR-10

Figure 5: **Training and test dynamics on easy and difficult subsets** of MNIST (with MLP) and CIFAR-10 (with VGG-11). The memorization scores are used to partition the dataset and the percentage in parentheses indicates the size of the subset. Difficult examples (red) is learnt simultaneously with easy examples. As a control, the dotted line in (a) and (b) indicates training on the training data with the difficult subset removed. Thus, we confirm that the improvement of difficult examples' training accuracy is not a side effect of the patterns learnt from easy examples.

Figure 6(*a*) that training on the easy subset enables a fast convergence on the corresponding training accuracy, but a marginal improvement on that of the difficult subset. While in Figure 6(*b*), despite the relatively slow convergence on the difficult subset, the training benefits both the difficult subset and the easy subset substantially. It proves that the memorization of difficult examples improves the model's ability to generalize on easy examples, while the generalizable patterns learnt from easy examples do not transfer to difficult examples.

A difficult but correct example contains some (relatively weak) degree of similarity with the easy examples, and it results in a slow but still increasing training accuracy on the easy subset when the model is trained on the difficult subset. A difficult subset contains more (atypical and exclusive) subpopulations than an easy subset and thus is more **diverse**, so a model that is trained on an easy subset is very unlikely to acquire this unique information. Therefore, we conclude that difficult examples are more informative and to memorize difficult examples accelerates and benefits the model's ability to generalize.



(a) Train on an easy subset. (b) Train on a difficult subset.

Figure 6: **Training dynamics on two disjoint subsets of CIFAR-10 with VGG-11**. (a) shows the training accuracy when trained only with the easy subset. (b) shows the training accuracy when trained only with the difficult subset. Difficult examples contain information about the easy ones while the easy examples has very little information about the difficult ones.

4. Conclusion

Our work studies memorization regarding optimization in the context of random data and real data. We use turn-over dropout to efficiently evaluate memorization scores of training examples, and we unify random data with real data by showing that examples with random labels are simply extremely difficult examples. We find there are qualitative differences between the easy and difficult examples, and it is consistent for random data and real data that easy and difficult examples are learnt simultaneously, with difficult ones learnt slower. On real data, we also conclude that difficult examples contain more information than easy ones, and to memorize difficult examples may improve the model's ability to generalize, which may inspire core-set selection or more efficient training algorithms as future work.

References

Devansh Arpit, Stanisław Jastrzębski, 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.

Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 123–132, 2021.

Satrajit Chatterjee. Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. In *ICLR*, 2020.

Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. In *ICLR*, 2020.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

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.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In *NeurIPS*, 2020.

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

Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

Sosuke Kobayashi, Sho Yokoi, Jun Suzuki, and Kentaro Inui. Efficient estimation of influence of a training instance. 2020.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Hartmut Maennel, Ibrahim Alabdulmohsin, Ilya Tolstikhin, Robert JN Baldock, Olivier Bousquet, Sylvain Gelly, and Daniel Keysers. What do neural networks learn when trained with random labels? In *NeurIPS*, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

Appendix A. Experiment Set-up

For MNIST (Deng (2012)), we use a one-hidden layer MLP with 4096 neurons in the hidden layer trained with a 0.06 learning rate on 200 epochs and no momentum, except for Figure 3, in which we switch to a learning rate of 0.01 for clearer result. For CIFAR-10 (Krizhevsky et al. (2009)), we use a VGG-11 network (Simonyan and Zisserman (2015)) without batch-norm, and it is trained on 200 epochs, with the learning rate starts at 0.01 (with 0.9 momentum) and divided by 10 at epoch 100 and epoch 150 respectively. Additionally, spatial turn-over dropout is applied to VGG-11 on its convolutional layers. The batch size is 256 for training and all the models use the ReLU activation.

Appendix B. Implementation of Turn-over Dropout

For linear layers, turn-over dropout can be efficiently implemented to mask the neurons for two sub-networks. For convolutional layers, 2D spatial turn-over dropout is implemented, which deterministically masks filters (feature maps) instead of neurons. Half of the feature maps will be set to zero, and the other half will be passed to the next layer.

Appendix C. Easy and Difficult Examples Identified by Turn-over Dropout

Figure 7 and Figure 8 show the easy (low memorization score) and difficult (high memorization score) examples that are identified with turn-over dropout on MNIST and CIFAR-10 respectively. The easy examples do follow a typical pattern and shares a lot of similarities, and both the unmuted sub-network and the muted sub-network agree with the ground truth. The difficult examples seem to have some peculiar traits.

Appendix D. Early Stop

Figure 9 shows the training accuracy of the muted and unmuted sub-networks in a model with turn-over dropout. Both the muted and unmuted sub-networks converge very quickly within the first a few epochs and the stabilize. This motivates us to train the model with turn-over dropout for only a few epochs to calculate the memorization scores.

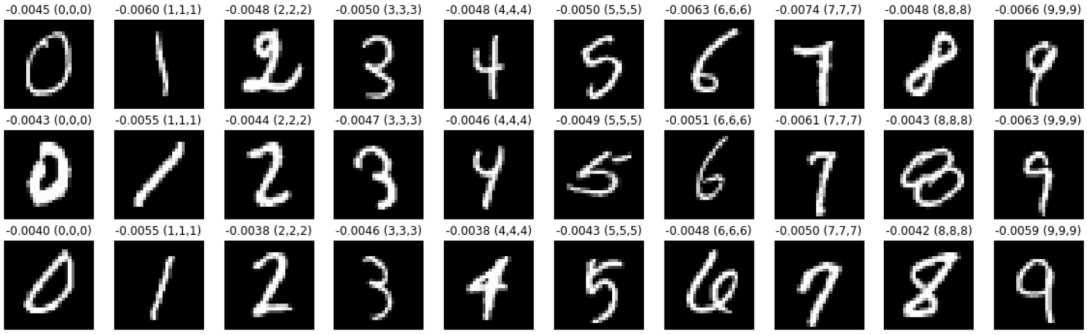
Appendix E. Training and Test Dynamics of Different Degree of Corruption

Figure 10 shows the training and test dynamics of a VGG-11 network trained on CIFAR-10, of which the labels of the training dataset are corrupted by different extent and the test dataset remains unchanged.

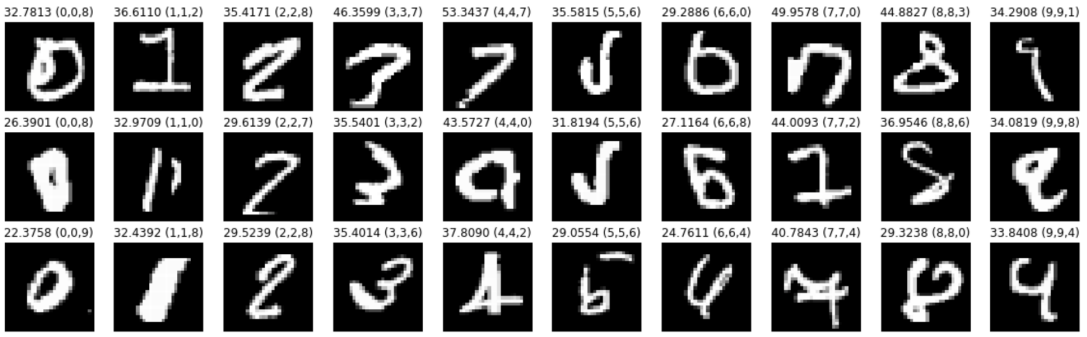
It can be observed that there are qualitative differences between the learning curves of different degree of corruption, which indicates that DNNs conduct memorization in a data-aware fashion. There are two main observations:

1. Difficult instances (random noises among the labels) make the learning process slower. The more difficult examples, the longer it takes to fully fit the dataset.

2. When there exists noise in the dataset, the test accuracy firstly reaches to a peak before it drops and converges.



(a) Low memorization score examples on MNIST dataset.



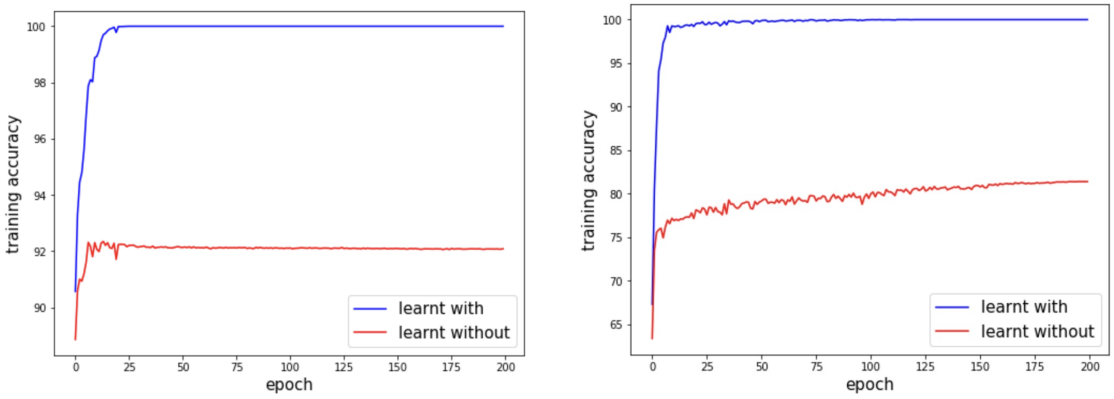
(b) High memorization score examples on MNIST dataset.

Figure 7: Low and high memorization score examples on MNIST dataset. Each column corresponds to one class. The number above the image represents the memorization score and the triplet above the image represents (ground truth label, unmuted sub-network's prediction, muted sub-network's prediction).

(a) Low memorization score examples on CIFAR-10 dataset.

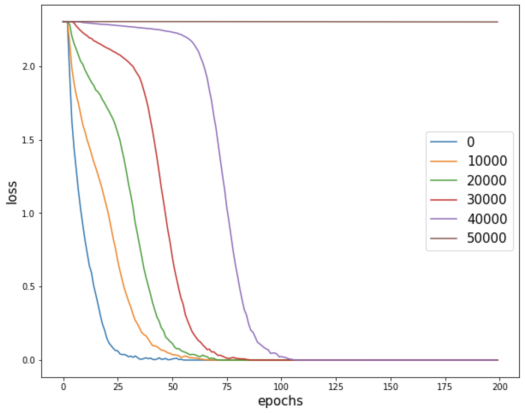
(b) High memorization score examples on CIFAR-10 dataset.

Figure 8: Low and high memorization score examples on CIFAR-10 dataset. Each column corresponds to one class. The number above the image represents the memorization score and the triplet above the image represents (ground truth label, unmuted sub-network's prediction, muted sub-network's prediction).

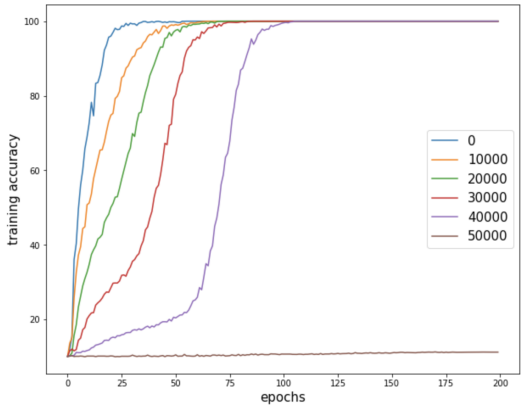


(a) MNIST. (b) Fashion MNIST.

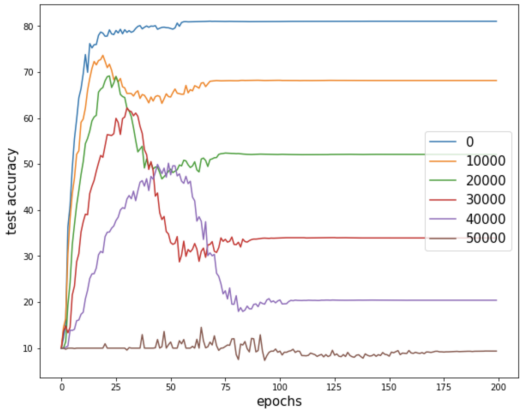
Figure 9: **Training accuracy of the muted and unmuted sub-networks in a model with turn-over dropout**. (a) a one-hidden-layer MLP trained on MNIST with turn-over dropout. (b) a compact vision transformer (Hassani et al., 2021) trained on fashion MNIST with turn-over dropout.



(*a*) Loss.



(*b*) Training accuracy.



(*c*) Test accuracy.

Figure 10: **Training and test dynamics of VGG-11 network on CIFAR-10 where the training dataset is corrupted by different extent and the test dataset remains unchanged.** The legend shows the number of examples with random labels. It should be mentioned that since we use a multi-step learning rate decay schedule instead of a cosine annealing schedule, the model becomes incompetent to fit the dataset when all labels are replaced by random noise. This also proves that the extent and capability of memorization is influenced by the training procedure as well.