Intersection of Parallels as an Early Stopping Criterion

Ali Vardasbi
University of Amsterdam
Amsterdam, The Netherlands
a.vardasbi@uva.nl

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

Mostafa Dehghani
Google Brain
Amsterdam, The Netherlands
dehghani@google.com

ABSTRACT

A common way to avoid overfitting in supervised learning is early stopping, where a held-out set is used for iterative evaluation during training to find a sweet spot in the number of training steps that gives maximum generalization. However, such a method requires a disjoint validation set, thus part of the labeled data from the training set is usually left out for this purpose, which is not ideal when training data is scarce. Furthermore, when the training labels are noisy, the performance of the model over a validation set may not be an accurate proxy for generalization. In this paper, we propose a method to spot an early stopping point in the training iterations of an overparameterized neural network (NN) without the need for a validation set. We first show that in the overparameterized regime the randomly initialized weights of a linear model converge to the same direction during training. Using this result, we propose to train *two parallel instances* of a linear model, initialized with different random seeds, and use their *intersection* as a signal to detect overfitting. In order to detect intersection, we use the cosine distance between the weights of the parallel models during training iterations. Noticing that the final layer of a NN is a linear map of pre-last layer activations to output logits, we build on our criterion for linear models and propose an extension to multi-layer networks, using the new notion of *counterfactual weights*. We conduct experiments on two areas that early stopping has noticeable impact on preventing overfitting of a NN: (i) learning from noisy labels; and (ii) learning to rank in information retrieval. Our experiments on four widely used datasets confirm the effectiveness of our method for generalization. For a wide range of learning rates, our method, called Cosine-Distance Criterion (CDC), leads to better generalization on average than all the methods that we compare against in almost all of the tested cases.

CCS CONCEPTS

• **Information systems** → **Learning to rank**.

KEYWORDS

Generalization, Overparameterization, Early stopping, Cosine distance

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557366

ACM Reference Format:
Ali Vardasbi, Maarten de Rijke, and Mostafa Dehghani. 2022. Intersection of Parallels as an Early Stopping Criterion. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557366

1 INTRODUCTION

Modern overparameterized neural networks can easily overfit to the training data, even when the data is noisy [22]. Collecting up-to-date and large-scale high accuracy labeled data is however expensive, time-consuming, and in many cases not feasible. The main question we address in this work is "how can we improve generalizability of overparameterized neural networks in low data regimes and noisy labeled data?"

Early stopping. Early stopping is one of the fundamental techniques for generalization in neural networks (NNs) with iterative training [14, 26]. The common practice for early stopping is to monitor model performance on a held-out set, called validation set, and quit training if the validation performance degrades or has not improved for several iterations. In such settings, the assumption is that model performance over the held-out data, which has not been seen by the model during training, is an unbiased proxy for the test performance. This poses a trade-off for choosing the size of the validation set: A small validation set would suffer from high variance in estimating the test performance, whereas a large validation set would reduce the size of the training set and possibly hurt the model generalizability. Additionally, it is known that overparameterized NNs can easily overfit noisy labeled datasets without early stopping [22]. However, as noted in [12], when training on a noisy labeled dataset, relying on an accurately labeled validation set for early stopping is not realistic.

Ideally, then, we would like to have a criterion for early stopping that does not rely on the existence of a validation set. Some validation set independent criteria have been proposed in the literature based on gradients [12, 24] or leave one out (LOO) interpolation [5].

Overparameterization. Despite the positive generalizability results of overparameterization that state that overparameterized models are less dependent on regularization [4, 36], early stopping has been shown to still be helpful in overparameterized models [21], especially in the presence of label noise [22].

A proposal for early stopping. In this work, we propose a new criterion for early stopping NNs with iterative optimization methods such as (stochastic) gradient descent, that is particularly helpful in low data regimes and noisy labeled datasets. We start with linear models and then describe how to extend our method to be effective for non-linear multi-layer networks as well. Our early stopping criterion is based on our experimental observation that different instances of an overparameterized linear model, initialized with

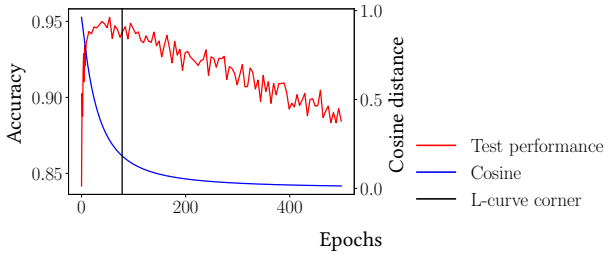

Figure 1: Convergence of cosine distance of parallel instances coincides with overfitting.

different random seeds, tend to converge to a similar solution on the fitness surface during training. Fig. 1 plots the cosine distance between the weight vectors of two linear networks as well as test performance during the training. First, we observe that the cosine distance forms an L-curve. We indicate the corner of this L-curve by a vertical black line. We hypothesize that when the cosine distance starts to converge to zero, i.e., the two parallel instances intersect, the models have already learned the "generalizable" patterns for solving the task and after that point they start picking up spurious features that only represent the training set. This is observable in Fig. 1, where the test performance continues to decrease after the L-curve corner. Our goal is to find the transition point between the two extreme cases: (i) The first training iterations where the parallel instances are far from each other. This case is captured by weight vectors that are randomly directed and are almost surely orthogonal. (ii) The converged iterations where the parallel instances become very close to each other, i.e., intersect. This case is captured by the weight vectors that are aligned with the direction of zero training loss. To this end, we measure the cosine distance between the weight vectors of parallel models at each iteration. Due to the two extreme cases mentioned above, the cosine distances with respect to training iterations form an L-curve (Fig. 1). We use two simple methods, namely the maximum curvature and a threshold-based criterion, to detect the transition point, and output the corresponding iteration as the early stopping point. Our experiments on a diverse set of datasets on different data modalities show that the transition point obtained as above for the early stopping iteration leads to better generalization results than existing methods.

Going beyond linear models. Our observation of converging parallel instances only holds for linear models where the weight vector is used to map a fixed input matrix to a fixed target vector. For networks with more than one layer and non-linear activation functions, however, the weight vector of the last layer maps the activation matrix of the previous layer to the target vector. So for two different randomly initialized networks $\mathcal{N}_1$ and $\mathcal{N}_2$, last layer activations, A_1 and A_2, may be in completely different spaces and not directly comparable. Consequently, the described L-curve criterion cannot be used as-is for the linear mapping in the last layer of networks with multiple layers. To remedy this, we first answer the counterfactual question of: *what would be the weight vector of $\mathcal{N}_2$ if its last layer activation A_2 was to be replaced with A_1 from $\mathcal{N}_1$?* Using a linear algebra trick, we find an answer for this counterfactual question, and as the settings have been made similar to the linear case, we observe a similar L-curve trend in the cosine distance between the weight vector of $\mathcal{N}_1$ and the counterfactual weight vector of $\mathcal{N}_2$. Our experiments on different datasets and with a wide range of learning rates confirm the effectiveness of this method on multi-layer networks.

Our contributions in this paper are:

(1) We empirically show that for overparameterized linear models, the randomly initialized weight vectors, when trained by an iterative method such as stochastic gradient descent (SGD), converge to the same solution.
(2) We use the above finding on linear models to detect an early stopping point without a validation set by measuring the cosine distance between the weight vectors of two parallel instances of a linear model, initialized with different random seeds. We call our method *Cosine-Distance Criterion* (CDC).
(3) Using the counterfactual weight vector of the last layer of one of the instances, we extend our proposed cosine distance criterion to multi-layer networks.
(4) We experimentally verify the generalization effectiveness of CDC on two widely used computer vision datasets with noisy labels as well as two widely used learning to rank (LTR) datasets. Particularly, we show that CDC leads to better average test performance, i.e., better generalization, than the methods against which we compare across a wide range of learning rates.

1.1 Notation

Let $X_{N\times d} = [x_1\ x_2\ \cdots\ x_N]^T$ denote the training data consisting of $x_i \in \mathbb{R}^d$ as feature vectors and let $Y_{N\times 1} = [y_1\ y_2\ \cdots\ y_N]^T$ denote the corresponding labels. For a one-layer network, i.e., a linear model, we use a (non-trainable) featurization function $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ with $D > N$ to have an overparameterized model. We use $A_{N\times D} = [\Phi(x_1)\ \cdots\ \Phi(x_N)]^T$ for the featurized inputs. Training a linear model means finding a weight vector $W = [w_1, w_2, \ldots, w_D]^T$ satisfying:

$$A \cdot W = Y. \tag{1}$$

For a m-layer network, layer i can be written as:

$$A_{i+1} = \sigma\left(A_i \cdot W_i\right) \tag{2}$$

where σ is the activation function, and $A_0 = X$.[1] Focusing on the last layer, training such a network means finding parameters leading to activation matrix A_m and a weight vector W_m satisfying:

$$A_m \cdot W_m = Y. \tag{3}$$

Here, as opposed to Eq. (1) where A only depends on the input data, A_m depends on the trainable parameters. This means that, for different initializations, A_m in Eq. (3) will be different, while A in Eq. (1) is fixed. In what follows, when no confusion is possible, we drop the subscripts and write A and W for the pre-last layer activation matrix and last layer weight vector of a multi-layer network, respectively. For both linear and multi-layer networks, we use subscripts to differentiate between instances that are initialized with different random seeds. We use A^+ for the Moore-Penrose inverse of matrix A.

Finally, we show gradient descent (GD) iterations by parenthesized superscripts. For example, iteration t of a multi-layer network

[1] Without loss of generality we assume the bias is zero. We can always concatenate a single 1 to the end of feature vector x to model the bias.

satisfies:

$$A^{(t)} \cdot W^{(t)} = \tilde{Y}^{(t)}, \tag{4}$$

where $\tilde{Y}$ is the model prediction. Iteration 0 stands for the initialization values.

2 RELATED WORK

Early stopping. The idea of early stopping in machine learning is based on the assumption that in the process of training iterations of a model, the model first learns the general relation between the inputs and labels, and then, gradually overfits to the training samples [26]. Various studies investigate the effect of early stopping or propose rules to identify early stopping points, especially with the gradient descent algorithm [21, 35]. A common practice for early stopping is to hold out part of the training data, usually referred to as the validation set, and train the model on the remaining data. The validation set is used to periodically evaluate the model during training, and the validation performance is considered as a generalization proxy.

As noted in previous work on early stopping [5, 24], using a validation set for early stopping has some drawbacks, especially in the low data regime. As the validation performance is going to be used as a proxy for generalization, its size should not be small. On the other hand, when the annotated data is already small, consuming a considerable part of it for the validation set, reduces the size of the training data and can lead to degraded performance. Relying on the validation set is also not desired for noisy labeled data where the performance on a noisy labeled validation set may not be a good proxy for the actual performance on unseen data [12].

These limitations have led some researchers to propose early stopping rules without a validation set. Mahsereci et al. [24], building on [10], use gradients to identify the early stopping point. The idea is when the gradients become small, it is a signal indicating the model has learned the general structure, and it is starting to overfit to the train data. Similarly, Forouzesh and Thiran [12] use the gradients for an early stopping criterion. Instead of the size of the gradients, they measure the disparity of gradients, i.e., the effect of a gradient step on one mini-batch on another one, and decide to stop early when a number of training iterations with increased disparities are observed. Finally, Bonet et al. [5] use LOO interpolation to characterize generalization and propose to stop the training if for a pre-defined number of iterations the risk estimated from LOO interpolations is increased. As their method uses LOO samples, it can be thought of as an efficient modified cross validation (CV) with as many folds as the size of training data.

Our CDC method differs from previous work in that it uses two parallel instances of a model to detect early stopping. We show experimentally that the change of parameter vector direction during training of one network does not provide a helpful signal for overfitting. Furthermore, instead of monitoring the gradients of the parameters, we track the parameters themselves. Parameter values at each iteration depend on their initial values, their gradients, and the learning rate. As such, our method can better adapt to different learning rates compared to previous work that uses gradients for their early stopping rule. We will verify this intuition in our experiments. Lastly, CDC is agnostic to the optimization method used, so it is the same for gradient descent and stochastic gradient descent methods (unlike [24]) and does not depend on the mini-batch size or the subset size of mini-batches to approximate gradient discrepancy (unlike [12]).

Li et al. [22] analyze the robustness of neural networks to noisy labels, when trained with gradient descent. They show that overparameterized networks with early stopping, where the parameters are still close to their initial values, can robustly learn the correct labels and ignore the noisy ones. But after many more iterations, where the model goes far from its initialization, overfitting to the noisy labels occurs. The intuition of our CDC method is very close to that work in the sense that we stop training as soon as the parameters become far from their initial value. The difference is that we use two parallel instances of a model to estimate the point with lowest generalization risk. Using two instances allows us to spot the desired point even with large learning rates, where the model overfits soon and looking at the distance of one model to its initialization point is not enough (see Sec. 3.2).

Overparameterized models. A model is said to be *overparameterized* when its number of trainable parameters is bigger than the number of training samples. Modern machine learning models are usually highly overparameterized with good generalization properties. In order to better understand overparameterized neural networks, various studies analyze two-layer networks and examine their generalization characteristics [2, 23, 32, 37]. Overparameterization is essential for this work, because only in this case the gradient descent converges to the solution with minimum ℓ_2 distance from the initialization point [11]. Since we base our early stopping rule on the convergence of weights of linear models, our rule only works with overparameterized models. In modern machine learning, this is not a concern, as almost all the leading models are overparameterized.

A number of studies analyze overparameterized linear models, or two-layer wide networks with only the last layer trainable [4, 28]. Muthukumar et al. [27] and Kini and Thrampoulidis [19] analyze the impact of loss on generalization of linear overparameterized models. Montanari and Zhong [25] characterize the generalization error of minimum-ℓ_2 norm solution of linear models. There are also studies such as [3, 9, 33] that analyze gradient descent on linearly separable data with overparameterized linear models.

L-curve corner detection. Detecting the corner of an L-curve is a popular regularization method for solving systems of linear equations with ill-conditioned matrices [15]. There are several advanced methods to solve this problem, [e.g., 7, 18], but for the purpose of our work, we adopt two basic methods: (i) The maximum curvature point [15]; and (ii) A fixed threshold on the cosine distance as an indication of the transition from generalization to overfitting. We show with extensive experiments that these two basic methods work fine in detecting the transition point for early stoping.

3 Cosine-Distance Criterion

The Cosine-Distance Criterion (CDC) for early stopping is based on the convergence of weight vectors of two instances of a linear model that are initialized with different random seeds. In this section, we first discuss this convergence behavior and then show how it can be used for early stopping in a linear model. Finally, we present an extension of CDC to multi-layer networks.

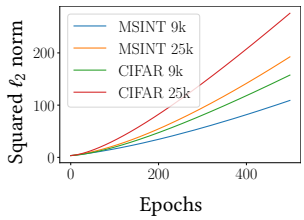

Figure 2: The growth of the weight vectors ℓ_2 norms for MNIST and CIFAR datasets and hidden layer widths of 9k and 25k.

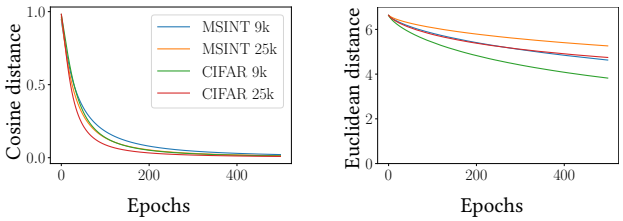

Figure 3: Convergence of the direction (left) and magnitude (right) of randomly initialized weight vectors of two identical but differently initialized models for MNIST and CIFAR datasets and hidden layer widths of 9k and 25k.

3.1 Convergence of Weights in Linear Model

In an overparameterized system, there are infinite directions of weight vectors that minimize the training loss. In the case of the RMSE loss, it can be shown that gradient descent converges to the solution with minimum ℓ_2 distance from the initialization point [11]. The common practice in current neural models is to use the Xavier initialization [13] that gives rise to $\mathbb{E}\left[|W^{(0)}|^2\right] = \frac{1}{3}$. As proved in [22] for a wide class of datasets, including noisy labeled datasets, network weights should converge to large values so that the network is able to distinguish inputs with small difference, but different labels. Our empirical results confirm this for all of our tested datasets. For example, Fig. 2 illustrates the growth of ℓ_2-norms of linear models on MNIST and CIFAR datasets with different widths for the hidden layer (9k and 25k). Consequently, the ℓ_2-norm of the initial weights becomes negligible compared to the ℓ_2-norm in the higher iterations. This means that the ℓ_2-norm of the converged weights becomes dominant and the ℓ_2 distance from the initialization point can be approximated by the ℓ_2-norm itself. In other words,

> Given two instances of an overparameterized linear model that are initialized using Xavier initialization, but with different random seeds, training them using *SGD leads to similar solutions on the fitness surface.*

Fig. 3 illustrates examples of this result. The left plot shows the trend of the cosine distance between the weight vectors of two parallel instances of a linear model, initialized with different random seeds, in terms of gradient descent epochs. Similar to Fig. 2, the results are reported for MNIST and CIFAR with hidden layer widths of 9k and 25k. We repeat the experiment for different learning rates and different model sizes and consistently observe that the cosine distance always converges to zero with rates depending on the learning rate and model size. This observation also holds for the two LTR datasets that we consider in this paper (see Sec. 4).

Start of overfitting. In overparameterized linear models, the exact minimum-ℓ_2 solution can be obtained using the Moore-Penrose inverse [30]. This exact solution corresponds to one of the zero training loss solutions of the model. Our experiments show that, although GD converges to the minimum-ℓ_2 solution, it requires significantly longer training epochs than what is observed in Fig. 3-left. For instance, after 100k training epochs, the cosine distance between the GD and minimum-ℓ_2 solutions in different data and different model setups falls in the range of 0.1 and 0.3. Comparing to Fig. 3, where at epoch 400 the weights of two parallel instances have a cosine distance less than 0.01, we see a large gap between the convergence rates of these two cases:

> *The parallel instances converge to each other long before they converge to their final value of the minimum-ℓ_2 solution.*

Noticing that overfitting can start at sooner iterations than the zero training loss point, we conjecture that this early convergence coincides with the start of overfitting to the training data and provide evidence for our conjecture with a diverse set of experiments.

Formal notation. Our discussion so far can be summarized in the following formal notation. Suppose two parallel instances of a linear network $\mathscr{L}_1$ and $\mathscr{L}_2$, with different initializations $W_1^{(0)}$ and $W_2^{(0)}$, featurized input A and target output Y. Both models will converge to the minimum ℓ_2 zero training loss solution, obtained by $W^{(\infty)} = A^+ \cdot Y$, but, based on our empirical findings, $W_1^{(t)}$ and $W_2^{(t)}$ converge in direction at a rate considerably faster than they converge to $W^{(\infty)}$. Intuitively, the reason for such a behavior is that the models first learn the general structure of the training data and ignore the outlier samples [22]. Suppose the (unknown) set of the non-outlier (featurized) training samples that best describe the general data structure and their corresponding clean labels are shown by $\hat{A}$ and $\hat{Y}$. Then, by learning the general structure, each model first converges to the $\hat{A}^+ \cdot \hat{Y}$ solution. A low distant pair of $W_1^{(t)}$ and $W_2^{(t)}$ vectors indicates that the general $\hat{A}^+ \cdot \hat{Y}$ solution has been reached and, from now on, the models are converging to the zero training loss solution of $A^+ \cdot Y$.

Directions or magnitudes. Finally, to show that the above trends are only observable for the *directions* and not the *magnitudes*, we replace the cosine distance with the Euclidean distance in the right plot of Fig. 3. As can be seen in this figure, the Euclidean distance does not converge to zero within a practical number of epochs. More importantly, when the model overfits (which is the case in all of the shown plots), the Euclidean distance does not converge to a fixed value across different datasets, model sizes and learning rates, making it difficult to base an early stopping criterion on it.

3.2 Cosine-Distance Criterion

Based on what we have discussed in the previous section, we can state our simple rule for early stopping:

> *Train in parallel two identical instances of a linear model, initialized with different random seeds, and stop training when the L-curve obtained from the cosine distance between their weight vectors in terms of training epochs passed its corner.*

Findings of Li et al. [22] (as discussed in Sec. 2) support this idea to

some extent. They show that in the presence of corrupted labels, the first iterations of gradient descent ignores them and tries to fit on the correct labels. During this phase, the weights remain close to their initial values. But as the iterations proceed, the model starts to overfit the corrupted labels and the weights move far from their initial value. Here, our empirical observation suggests a more specific behavior: networks converge to directions far from their initialization, but this new direction is the same for identical but differently initialized networks and it is not necessarily the direction corresponding to zero training loss.

Using parallel instances. In order to materialize this idea for early stopping, we need a robust method to detect when the weights of the linear model are *converged* far enough from their initial value, and the model is starting to overfit. Two extreme cases are clear: (i) at the beginning iterations, the weights are close to the initialization and the model underfits; but (ii) after too many iterations, the weights are far from the initialization and the model overfits. Knowing these two extreme cases is not enough. Instead, we are interested in finding the transition point between them. Looking at one model and measuring the distance of its weights to their initial value is not sufficient for detecting the transition point, as it is not clear *how far* from the initialization point the transition occurs. Our experiments reveal that this issue is solvable by comparing two identical but differently initialized models, instead of only looking at a single model. Two identical models, starting from two different initialization points, converge to the same zero training loss point. As a consequence of this shared final destination, they are converging to each other too. The closer they get to each other, they will be closer to their shared final destination of zero training loss, and they overfit more to the train data. Therefore, the similarity of these models at each iteration can be used as a proxy for detecting their degree of overfitting. Starting from a cosine distance close to one (as two random high dimensional vectors are almost surely orthogonal), the two models converge to each other as they learn the training data. After the general structure of the training data has been learned, the two models start to overfit. Our empirical observations on four datasets with different modalities and models with different sizes and learning rates suggest that the changing rate of the cosine distance differs before and after the sweet point of maximum generalization: the cosine distance decreases sharply before that point, but slowly after it. That is why we observe an L-curve similar to what was shown in Fig. 1. We take the corner of such an L-curve as the maximum generalization point.

Algorithm 1: CDC

Input: D, η, $T_{\max}$
Output: Early Stopping Epoch

1 Build $\mathcal{N}_1$ and $\mathcal{N}_2$ with weights W_1 and W_2 of length D
2 Initialize W_1 and W_2 with different random seeds
3 **for** $t = 0$ *to* $T_{\max}$ **do**
4 Update W_1 and W_2 using first order methods with learning rate η
5 $\delta^{(t)} = \cos(W_1^{(t)}, W_2^{(t)})$
6 **if** $\delta^{(t)}$ *is a L-curve corner* **then**
7 Return t
8 **end**
9 **end**
10 Return $T_{\max}$

The algorithm. Algorithm 1 shows the formal pseudo-code for CDC. The algorithm gets as input the number of parameters (D), learning rate (η) and the maximum number of epochs ($T_{\max}$). Despite its simplicity, our extensive experiments show the effectiveness of CDC compared to existing methods for a wide range of learning rates. For line 6 of Algorithm 1, we test two simple methods: (i) the maximum curvature condition; and (ii) a fixed threshold for all cases. Next, we discuss and compare these two methods for corner detection in more detail.

3.3 Corner Detection

As discussed in Sec. 2, the L-curve corner detection is a popular regularization method for solving systems of linear equations [7, 15, 18]. Here, we show our method has a low sensitivity to the hyper parameters of corner detection methods and in all of our tested datasets and model setups, it is safe to consider a fixed threshold for the cosine distance as the early stopping point.

Maximum curvature. Let c_i be the cosine distance between the two models at iteration i. Since we are working with a discrete series, we need a step size Δ to act as the sampling points in our curves: each c_k value corresponds to the virtual $(i \cdot \Delta, c_i)$ point in the 2-D plain. Using this terminology, curvature is defined as follows (adapting [15] to our terminology):

$$\kappa_i = \Delta \cdot \frac{c_i - c_{i-2}}{\left(\Delta^2 + (c_i - c_{i-1})^2\right)^{\frac{3}{2}}}. \tag{5}$$

Here, Δ is a hyper parameter, modeling the compression of the x-axis in plots similar to Fig. 1. In practice, the $\{c_i\}$ series may not be smooth. So, before computing Eq. (5), we first apply Gaussian kernel smoothing to the series. The iteration with maximum κ_i is returned as the corner point in line 6 of Algorithm 1. For the hyper parameter Δ, our experiments show a very low sensitivity as will be discussed next. Intuitively, the discretization step should be in the order of the learning rate. In Fig. 4 we show two examples on MNIST (left) and CIFAR (right) datasets. The solid vertical black line is the maximum curvature obtained by setting Δ equal to the learning rate (0.004 and 0.002 for MNIST and CIFAR, respectively). The shaded area contains the epochs with maximum curvature for different Δ values, ranging from $\frac{lr}{2}$ to $5 \times lr$. The test performance drops, for this wide range of hyper parameter, are only 97% and 93% for MNIST and CIFAR, respectively. Various experiments on other datasets (see Sec. 4) and with different model sizes and learning rates lead to similar results.

Fixed threshold. We use the cosine distance to measure the similarity of two identical but differently initialized instances of a model and want to decide about the starting point of overfitting from this similarity. Therefore, it is natural to set a fixed threshold for this similarity and early stop when the instances *intersect*, i.e. become more similar than that threshold. Here, it is not theoretically clear what choice for the fixed threshold leads to the optimal results and whether this threshold should depend on the dataset and the model setup or not. However, our empirical observations on four

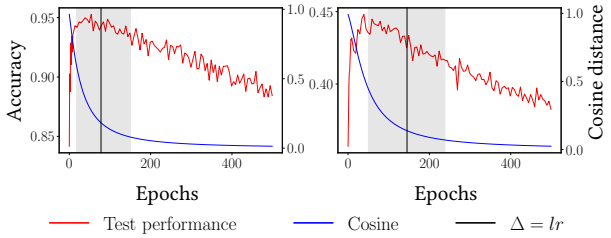

Figure 4: Sensitivity of the maximum curvature corner detection method to Δ on MNIST (left) and CIFAR (right) datasets with 50% random labels.

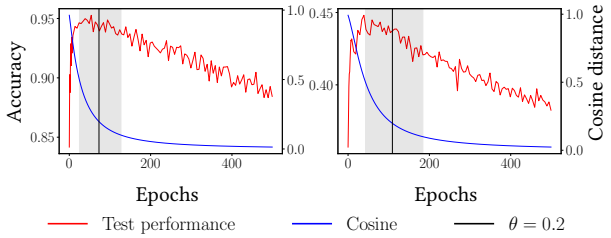

Figure 5: Sensitivity of fixed threshold corner detection method to θ on MNIST (left) and CIFAR (right) datasets with 50% random labels.

datasets from two different domains (i.e. vision and LTR) and on a wide range of model sizes and learning rates show that a threshold of $\theta = 0.2$ works fine in most of the cases. This threshold value also works for our multi-layer experiments (to be discussed later in Sec. 3.4). We leave further analysis of why a threshold of $\theta = 0.2$ for the cosine distance works fine for a very wide range of datasets and model setups to future work. Now, similar to what we have shown previously in the maximum curvature method, we show that our method has a low sensitivity to this θ hyper parameter. In Fig. 5 we show two examples on MNIST (left) and CIFAR (right) datasets. The solid vertical black line is where the cosine distance crosses $\theta = 0.2$. The shaded area contains the epochs with $0.1 \leq \theta \leq 0.5$. The test performance drops, for this wide range of hyper parameter, are only 98% and 94% for MNIST and CIFAR, respectively. Various experiments on other datasets (see Sec. 4) and with different model sizes and learning rate lead to similar results.

As the fixed threshold method has a lower sensitivity and it is simpler, we only report the performance of our criterion with this corner detection method and set $\theta = 0.2$. We should also stress that the maximum curvature method with $\Delta = lr$ leads to similar results.

3.4 Multi-Layer Networks

As discussed in Sec. 1.1, in a multi-layer network, the pre-last layer activation matrix depends on trainable parameters. As such, for two parallel instances $\mathcal{N}_1$ and $\mathcal{N}_2$ with different initializations, we have different, time-dependent activation matrices $A_1^{(t)}$ and $A_2^{(t)}$. The target vector Y, on the other hand, is shared for both instances. This means that $W_1^{(t)}$ and $W_2^{(t)}$ should map different inputs $A_1^{(t)} \neq A_2^{(t)}$ to a shared output. Consequently, comparing $W_1^{(t)}$ and $W_2^{(t)}$ is comparing apples and oranges. Between the two instances, only the predictions $\tilde{Y}_1^{(t)}$ and $\tilde{Y}_2^{(t)}$ are comparable, since they are both

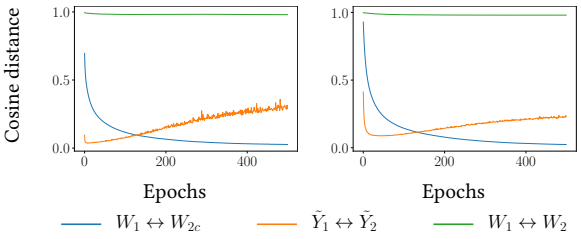

Figure 6: Cosine distance of different vectors for 2-layer networks on MNIST (left) and CIFAR (right) datasets with 50% random labels.

produced to match the same target label vector Y. In order to settle this issue, we try to answer the following counterfactual question:

> *Had $\mathcal{N}_2$ been given the activation matrix of $\mathcal{N}_1$, that is, $A_1^{(t)}$ instead of $A_2^{(t)}$, what would have been its weight vector to produce the prediction $\tilde{Y}_2^{(t)}$?*

We call $W_{2c}^{(t)}$ the *counterfactual weight* of $\mathcal{N}_2$ as the answer to the above question:

$$A_1^{(t)} \cdot W_{2c}^{(t)} = \tilde{Y}_2^{(t)}. \tag{6}$$

The best solution in terms of ℓ_2 error to Eq. (6) is given by the Moore-Penrose inverse:

$$W_{2c}^{(t)} = A_1^{(t)+} \cdot \tilde{Y}_2^{(t)}. \tag{7}$$

It is worth noting that $W_1^{(t)} = A_1^{(t)+} \cdot \tilde{Y}_1^{(t)}$. Or in words, $W_1^{(t)}$ and $W_{2c}^{(t)}$ are projections of predictions $\tilde{Y}_1^{(t)}$ and $\tilde{Y}_2^{(t)}$, using the same matrix of $A_1^{(t)+}$.

In Sec. 3.1 we argued that the networks first learn the general structure of data by fitting to the "set of the non-outlier samples that best describe the general data structure and their corresponding clean labels, shown by $\hat{A}$ and $\hat{Y}$". Here, in case of multi-layer networks, when $\mathcal{N}_1$ and $\mathcal{N}_2$ are fit to $\hat{Y}$, it means that $\tilde{Y}_1^{(t)}$ and $\tilde{Y}_2^{(t)}$ agree on the (unknown) $\hat{Y}$ part of their predictions. But, since they do not overfit yet, they have random predictions about the outliers. Consequently, measuring the distance between $\tilde{Y}_1^{(t)}$ and $\tilde{Y}_2^{(t)}$ is simply not informative. However, when they are projected using $A_1^{(t)+}$, they should be mapped to weight vectors close to $\hat{A}^+ \cdot \hat{Y}$.

We experimentally confirm the above theory by showing that the cosine distance between $\tilde{Y}_1^{(t)}$ and $\tilde{Y}_2^{(t)}$ diverges instead of converging, while their projections $W_1^{(t)}$ and $W_{2c}^{(t)}$ converge. Fig. 6 shows two examples of the results of our experiments on MNIST (left) and CIFAR (right) datasets. All the other setups for a 2-layer network lead to similar observations. Here, we plot the cosine distance between three pairs of vectors: (i) Weight vectors $W_1^{(t)}$ and $W_2^{(t)}$ are not comparable and their direction remains orthogonal during training. (ii) Predicted vectors $\tilde{Y}_1^{(t)}$ and $\tilde{Y}_2^{(t)}$ do not converge in direction, because $\mathcal{N}_1$ and $\mathcal{N}_2$ first try to fit to $\hat{Y}$ and predict differently for the other (noisy or outlier) part of the samples. (iii) Finally, the projected vectors $W_1^{(t)}$ and $W_{2c}^{(t)}$ are comparable and converge in direction, similar to what we observe in linear models (e.g., Fig. 1).

The limitation of this method is when $A_1^{(t)}$ is rank deficient, so the solution given by Eq. (7) is not correct. In our experiments, we encountered such situations for mildly overparameterized networks with four or more layers. We leave further investigations of when this happens and what can be done to fix it for future work.

4 EXPERIMENTS

In this section we explain the datasets and model used in our experiments. We also explain the runs that we use to compare and show the effectiveness of our CDC rule in generalization.

4.1 Setup

Image datasets. We use two image classification benchmark datasets: MNIST and CIFAR-10. Both datasets consist of 10 classes. MNIST contains 60k training and 10k test grayscale 28×28 images of handwritten digits, while CIFAR-10 has 50k training and 10k test RGB 32×32 object images. Following [12], we make these datasets noisy by changing 50% of the labels to random values.

LTR datasets. As a regular choice in LTR research [16, 17, 34], we use two popular LTR datasets: Yahoo! Webscope [8] and MSLR-WEB30k [31]. For each query, these datasets contain a list of documents with human-generated 5-level relevance labels. After cleaning, the query-document feature vectors of Yahoo! and MSLR are 501 and 131 in length, respectively. For our experiments we randomly select 100 and 30 training queries from Yahoo! and MSLR to have around[2] 3.3k training samples (i.e., document-query pairs) for each dataset. We repeated experiments with smaller selections (down to 330 training documents) and got similar results. For both datasets, we use the test set split as provided in the original data. The Yahoo! dataset contains 6.7k test queries and 163k test documents; MSLR contains 6.1k and 749k query and documents in its test set.

Model. For image classification tasks, we report results from overparameterized models with three different sizes: (i) **Small**; (ii) **Medium**; and (iii) **Large**. For linear models, we respectively use $9k$, $25k$, and $50k$ featurization, leading to $90k$, $250k$, and $500k$ parameters. For two-layer networks, we respectively use 150, 350, and 700 hidden layer widths, leading to $119k$, $278k$, and $556k$ parameters for MNIST and $462k$, $1M$, and $2M$ parameters for CIFAR-10 datasets. The last layer of image classification tasks has a dimension of $D \times 10$, because of the 10 classes. For our method, we work with *vectors*. As a work-around, we consider the $D \times 10$ matrix as 10 vectors of length D and take the average cosine distance between the 10 vector pairs of parallel instances. We use the cross-entropy loss for image classification.

For LTR tasks, we only report the results with $10k$ parameters due to space limitations. To show the effectiveness of our method with different loss functions, we report the results with two popular LTR loss functions, namely the pointwise RMSE loss and the listwise ListNet loss [6].

Metric. We report the classification accuracy on the test set for image classification tasks. For LTR methods, we evaluate models in terms of their NDCG@10 performance on the test set.

4.2 Baselines

We compare our CDC method with the following existing work:

(1) **Cross Validation (CV):** The traditional way for early stopping. For this baseline, we use 5-fold cross validation to approximate the generalization drop: if for five consecutive epochs the cross validation performance did not improve, we stop the training.
(2) **Evidence-Based (EB):** The method proposed in [24] that uses the gradient size to estimate the transition point in the generalization curve.
(3) **Gradient Disparity (GD):** A recent method proposed in [12] based on gradient disparity. If the gradient disparity is increased for five consecutive epochs, the training is stopped.
(4) **Oracle:** The skyline performance obtained by monitoring the test performance for 500 epochs and selecting the epoch with maximum test performance. We choose the maximum of 500 epochs because in the settings of our experiments the maximum test performance occurs well before 500 epochs with probability almost equal to one. Note that this is not a realistic run as we do not have access to test labels in reality. We include this to compare the gap between different runs and the ideal case.

[2]The exact number is different for different random selections.

5 RESULTS

In this section we present our experimental findings by comparing CDC with other baselines (Sec. 4.2) using both overparameterized linear and two-layer networks.

5.1 Performance Comparison of Linear Models

First, we show for a wide range of learning rates, that CDC leads to near perfect generalization as opposed to other baselines. Fig. 7 shows the performance comparison of CDC with other baselines on image classification tasks for linear models with different sizes: small, medium and large as discussed in Sec. 4. In each plot, we compare the generalization performance for a wide range of learning rates. These plots show that, for almost all the learning rates, CDC is better able to detect the start of overfitting and stop training, compared to other baselines. Importantly, the gap between CDC and Oracle is negligible in most cases, meaning that it is hard to improve our stopping rule for the tested cases. As the learning rate is increased, degradation of test performance occurs more rapidly, making it more important and difficult to early stop at the correct iteration. This is observed in these plots as the performance of different methods diverge for higher learning rates. Even for this rather difficult task, we see CDC performs robustly well and close to the Oracle.

Fig. 8 contains a performance comparison of CDC with other baselines on the LTR datasets with RMSE and ListNet loss functions with a wide range of learning rates. Here we observe similar results: CDC is more robust in finding the correct generalizable point in all learning rates compared to other baselines and it performs very close to the skyline Oracle case.

A stopping rule that can have a performance very close to the oracle performance, reduces the sensitivity of the model to the choice of learning rate. This means that (i) hyper-parameter tuning for learning rate would be of less concern; and (ii) relatively larger learning rates can be chosen for the model to converge faster; hence, significantly reducing the training time.

For a quantitative comparison between different stopping methods, in Table 1 we compare the mean and standard deviations of stopping methods across different setups (i.e., data points from Fig. 7, 8 and 9). This table shows that in five out of six cases, our CDC performs significantly better than other methods on average. It also has the least or second least variance for most of the cases.

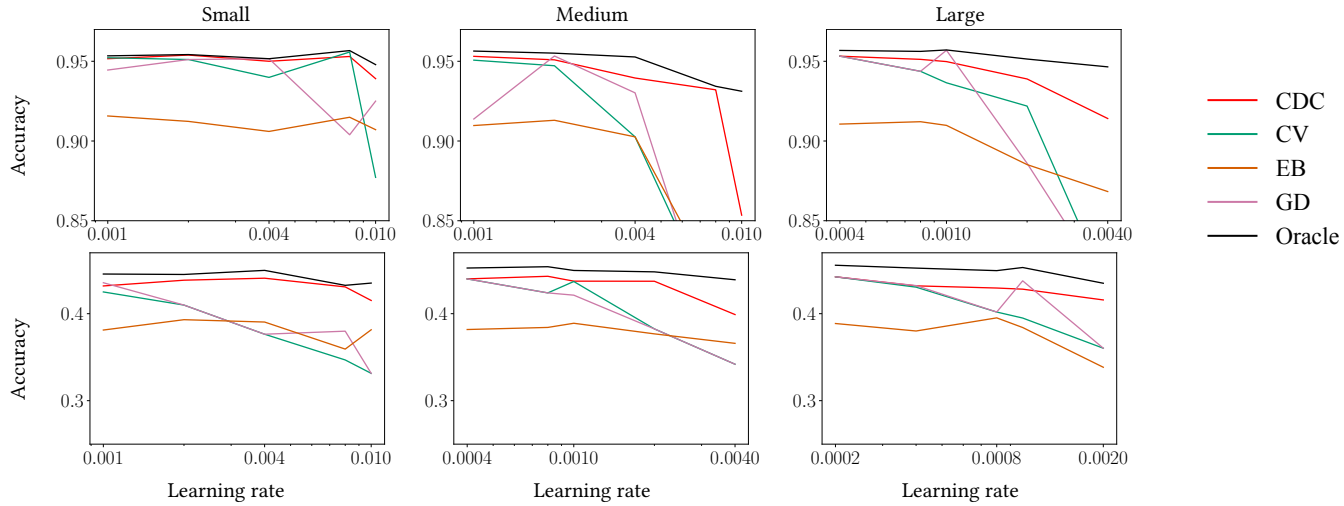

Figure 7: *Linear models*. Performance comparison of CDC with baselines in terms of test accuracy for linear models with different model sizes. Top: MNIST; Bottom: CIFAR-10; both with 50% random labels.

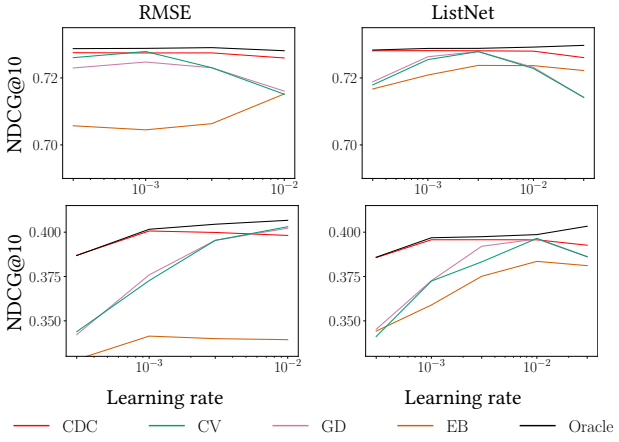

Figure 8: *Linear models*. Performance comparison of CDC with baselines in terms of test NDCG@10 for different LTR loss functions. Top: Yahoo! dataset; Bottom: MSLR dataset.

These results further show the reliability of CDC in different data modalities.

5.2 Two-Layer Networks

Similar to results in the previous section on linear models, in this section the generalization performance of CDC with other baselines on two-layer networks is examined. Fig. 9 contains the test performance of different baselines in terms of learning rates. We observe similar results to the linear case here: CDC performs more robustly on different learning rates compared to other baselines and it matches the Oracle performance, especially for medium and large networks.

The data points of these plots are aggregated and shown in Table 1 under columns indicated by "2-layer". In this table we observe that for the MNIST dataset, CDC and CV perform equally good and better than the other two baselines, while for the CIFAR-10 dataset, CDC is the sole winner.

5.3 Discussion

Our proposed method to find an early stopping point is based on detecting the intersection of two parallel instances of models. To materialize such a detection, we use the cosine distance of the weight vectors of parallel instances. This method of cosine distance works for linear models where all the learning is carried out by a single vector. One use case of our criterion can be in linear probing as one of the popular methods for transferring a pre-trained model to a downstream task [1, 20, 29]. With linear probing, we only train a linear layer that maps the representation from a frozen pre-trained backbone to the label space of the downstream task. Moreover, in general, in transfer learning, the downstream task has low training data. This lets CDC to be a great choice for early stopping in linear probing setup and can lead to better generalization than the existing methods.

Going from one layer to more layers with non-linearities poses a challenge on how to effectively detect the intersection point. Our proposed method in Sec. 3.4 considers the last layer of the instances to be able to continue to rely on the simple cosine distance. Our extensive experiments confirm that by this extension our method will be as effective on two-layer networks as it is on linear models. However, as the network gets deeper and more complex, the training dynamics might change and the last layer may not be sufficient to detect intersection of parallel instances. Consequently, our cosine distance criterion works best for linear and two-layer networks, and possibly for linear probing a pre-trained model. We believe that the general idea of this paper, i.e., using the intersection of two parallel instances of a model to signal early stopping, can be applied to more complex models, but with a more complex criterion that involves the whole model and not just the last layer. We leave further development of more complex intersection detection methods for future work.

6 CONCLUSION AND FUTURE WORK

We have proposed the Cosine-Distance Criterion (CDC) for early stopping in linear and two-layer networks. Our stopping rule does

Table 1: Mean and standard deviation of test accuracy and NDCG@10 across different model setups (Fig. 7, 8 and 9). Bold and underlined entries indicate best and runner up values. Superscripts * indicate significance improvements over the next best score with $p < 0.01$.

	MNIST				CIFAR-10				Yahoo!		MSLR	
	Linear		2-layer		Linear		2-layer		Linear		Linear	
	Mean	*STD*	Mean	*STD*	Mean	*STD*	Mean	*STD*	Mean	*STD*	Mean	*STD*
CDC	**0.939**	***0.0260***	0.947	*0.0069*	**0.431***	***0.0123***	**0.445***	*0.0093*	**0.727***	***0.0035***	**0.394***	***0.0137***
CV	0.910	*0.0586*	**0.947**	*0.0053*	0.396	*0.0380*	0.381	*0.0618*	0.722	*0.0067*	0.377	*0.0260*
EB	0.894	*0.0321*	0.910	***0.0024***	0.379	*0.0148*	0.399	***0.0042***	0.716	*0.0087*	0.354	*0.0231*
GD	0.902	*0.0704*	0.928	*0.0358*	0.401	*0.0370*	0.420	*0.0320*	0.722	*0.0071*	0.378	*0.0241*
Oracle	0.951	*0.0080*	0.953	*0.0020*	0.446	*0.0076*	0.451	*0.0059*	0.729	*0.0033*	0.398	*0.0130*

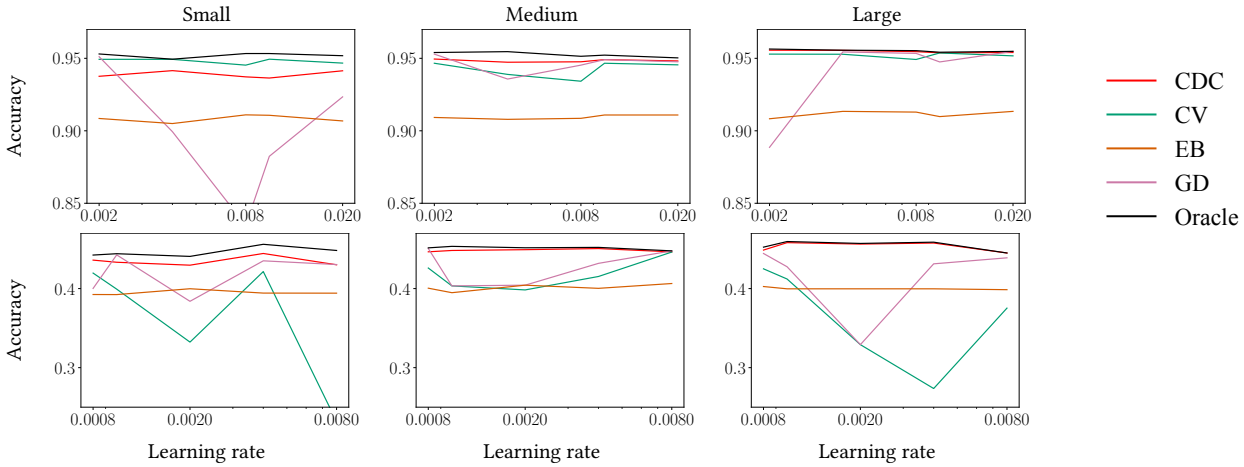

Figure 9: *Two-layer networks.* Performance comparison of CDC with baselines in terms of test accuracy for two-layer networks with different model sizes. Top: MNIST; Bottom: CIFAR10; both with 50% random labels.

not depend on a separate validation, nor does it use cross validation. This means the entire labeled data can be used for training. CDC is based on the consistent observation that two parallel instances of a linear model, initialized with different random seeds, will converge to the same solution on the fitness surface. This observation, together with supporting evidence from [22] on the distance of overfit weights to the initialized values of a network, led us to propose an early stopping rule based on the cosine distance of two parallel instances of an overparameterized linear model. The intuition is that when two randomly initialized weights start to become aligned, the parallel instances will intersect, and this is a signal for overfitting.

We have compared the generalization of our CDC rule with existing methods, namely cross validation, evidence-based [24] and gradient disparity [12], and shown that in all of the tested datasets and all model setups, CDC performs more reliably with different learning rates and there is a small gap between its performance and the skyline Oracle performance. We have also extended our method to work with multi-layer networks, using our notion of counterfactual weights vector. We have shown theoretically that our proposal is equivalent to comparing the projection of the output vectors of two instances, using a special projection matrix. Importantly, we have argued why the output vectors themselves cannot be used to detect the intersection point and verified our arguments experimentally.

The most interesting future direction to this work is to verify our conjecture about the intersection of parallel instances as the start of overfitting. Furthermore, finding a more complex criterion than the cosine distance to detect this intersection that involves the whole model and not just the last layer, and is suitable for more complex structures as well, would be a natural follow up to this work.

CODE AND DATA

To facilitate the reproducibility of the reported results, this work only made use of publicly available data and our experimental implementation is publicly available at https://github.com/AliVard/CDC-Early-Stopping.

ACKNOWLEDGMENTS

This research was supported by Elsevier and the Netherlands Organisation for Scientific Research (NWO) under project nr 612.-001.551, and by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. 2021. Exploring the Limits of Large Scale Pre-training. *arXiv preprint arXiv:2110.02095* (2021).

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. 2019. Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers. *Advances in neural information processing systems* (2019).

[3] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. 2018. A Convergence Analysis of Gradient Descent for Deep Linear Neural Networks. *arXiv preprint arXiv:1810.02281* (2018).

[4] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. 2019. Reconciling Modern Machine-learning Practice and the Classical Bias-Variance Trade-off. *Proceedings of the National Academy of Sciences* 116, 32 (2019), 15849–15854.

[5] David Bonet, Antonio Ortega, Javier Ruiz-Hidalgo, and Sarath Shekkizhar. 2021. Channel-Wise Early Stopping without a Validation Set via NNK Polytope Interpolation. *arXiv preprint arXiv:2107.12972* (2021).

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[7] J Longina Castellanos, Susana Gómez, and Valia Guerra. 2002. The Triangle Method for Finding the Corner of the L-curve. *Applied Numerical Mathematics* 43, 4 (2002), 359–373.

[8] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. *Journal of Machine Learning Research* 14 (2011), 1–24.

[9] Zeyu Deng, Abla Kammoun, and Christos Thrampoulidis. 2019. A Model of Double Descent for High-dimensional Binary Linear Classification. *arXiv preprint arXiv:1911.05822* (2019).

[10] David Duvenaud, Dougal Maclaurin, and Ryan Adams. 2016. Early Stopping as Nonparametric Variational Inference. In *Artificial Intelligence and Statistics*. PMLR, 1070–1077.

[11] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. 1996. *Regularization of Inverse Problems*. Vol. 375. Springer Science & Business Media.

[12] Mahsa Forouzesh and Patrick Thiran. 2021. Disparity between Batches as a Signal for Early Stopping. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 217–232.

[13] Xavier Glorot and Yoshua Bengio. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT press.

[15] Per Christian Hansen. 1998. *Rank-deficient and Discrete Ill-posed Problems: Numerical Aspects of Linear Inversion*. SIAM.

[16] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions. In *Proceedings of the 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 15–24.

[17] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 781–789.

[18] Jong-Dae Kim and Jong-Won Kim. 2007. Regularization Parameter Determination for Optical Flow Estimation using L-curve. *The KIPS Transactions: PartB* 14, 4 (2007), 241–248.

[19] Ganesh Ramachandra Kini and Christos Thrampoulidis. 2020. Analytic Study of Double Descent in Binary Classification: The Impact of Loss. In *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2527–2532.

[20] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. 2022. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054* (2022).

[21] Ilja Kuzborskij and Csaba Szepesvári. 2021. Nonparametric Regression with Shallow Overparameterized Neural Networks Trained by GD with Early Stopping. In *Conference on Learning Theory*. PMLR, 2853–2890.

[22] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. 2020. Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks. In *International conference on artificial intelligence and statistics*. PMLR, 4313–4324.

[23] Yuanzhi Li and Yingyu Liang. 2018. Learning Overparameterized Neural Networks via Stochastic Gradient Descent on Structured Data. *Advances in Neural Information Processing Systems* (2018).

[24] Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. 2017. Early Stopping without a Validation Set. *arXiv preprint arXiv:1703.09580* (2017).

[25] Andrea Montanari and Yiqiao Zhong. 2020. The Interpolation Phase Transition in Neural Networks: Memorization and Generalization under Lazy Training. *arXiv preprint arXiv:2007.12826* (2020).

[26] Grégoire Montavon, Geneviève Orr, and Klaus-Robert Müller. 2012. *Neural Networks: Tricks of the Trade*. Vol. 7700. Springer.

[27] Vidya Muthukumar, Adhyyan Narang, Vignesh Subramanian, Mikhail Belkin, Daniel Hsu, and Anant Sahai. 2021. Classification vs Regression in Overparameterized Regimes: Does the Loss Function Matter? *Journal of Machine Learning Research* 22, 222 (2021), 1–69.

[28] Vaishnavh Nagarajan and J Zico Kolter. 2019. Generalization in Deep Networks: The Role of Distance from Initialization. *arXiv preprint arXiv:1901.01672* (2019).

[29] Matthew E Peters, Sebastian Ruder, and Noah A Smith. 2019. To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks. *arXiv preprint arXiv:1903.05987* (2019).

[30] K Manjunatha Prasad and RB Bapat. 1992. The Generalized Moore-Penrose Inverse. *Linear Algebra Appl.* 165 (1992), 59–69.

[31] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).

[32] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. 2018. Theoretical Insights into the Optimization Landscape of Over-parameterized Shallow Neural Networks. *IEEE Transactions on Information Theory* 65, 2 (2018), 742–769.

[33] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. 2018. The Implicit Bias of Gradient Descent on Separable Data. *The Journal of Machine Learning Research* 19, 1 (2018), 2822–2878.

[34] Ali Vardasbi, Harrie Oosterhuis, and Maarten de Rijke. 2020. When Inverse Propensity Scoring does not Work: Affine Corrections for Unbiased Learning to Rank. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. ACM, 1475–1484.

[35] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. 2007. On Early Stopping in Gradient Descent Learning. *Constructive Approximation* 26, 2 (2007), 289–315.

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

[37] Mo Zhou, Rong Ge, and Chi Jin. 2021. A Local Convergence Theory for Mildly Over-parameterized Two-layer Neural Network. *arXiv preprint arXiv:2102.02410* (2021).