

**DEUTSCHES ELEKTRONEN-SYNCHROTRON**  
Ein Forschungszentrum der Helmholtz-Gemeinschaft



DESY 22-016  
arXiv:2201.12803  
January 2022

**Similarity and Generalization:  
From Noise to Corruption**

N. Fonseca

*Rudolf Peierls Centre for Theoretical Physics, University of Oxford, UK*

V. Guidetti

*Deutsches Elektronen-Synchrotron DESY, Hamburg*

ISSN 0418-9833

**NOTKESTRASSE 85 - 22607 HAMBURG**

DESY behält sich alle Rechte für den Fall der Schutzrechtserteilung und für die wirtschaftliche Verwertung der in diesem Bericht enthaltenen Informationen vor.

DESY reserves all rights for commercial use of information included in this report, especially in case of filing application for or grant of patents.

Herausgeber und Vertrieb:

---

**Verlag Deutsches Elektronen-Synchrotron DESY**

DESY Bibliothek  
Notkestr. 85  
22607 Hamburg  
Germany

---

# Similarity and Generalization: From Noise to Corruption

---

Nayara Fonseca<sup>1</sup> \*    Veronica Guidetti<sup>2</sup> \*

## Abstract

Contrastive learning aims to extract distinctive features from data by finding an embedding representation where similar samples are close to each other, and different ones are far apart. We study generalization in contrastive learning, focusing on its simplest representative: Siamese Neural Networks (SNNs). We show that Double Descent also appears in SNNs and is exacerbated by noise. We point out that SNNs can be affected by two distinct sources of noise: Pair Label Noise (PLN) and Single Label Noise (SLN). The effect of SLN is asymmetric, but it preserves similarity relations, while PLN is symmetric but breaks transitivity. We show that the dataset topology crucially affects generalization. While sparse datasets show the same performances under SLN and PLN for an equal amount of noise, SLN outperforms PLN in the overparametrized region in dense datasets. Indeed, in this regime, PLN similarity violation becomes macroscopical, corrupting the dataset to the point where complete overfitting cannot be achieved. We call this phenomenon *Density-Induced Break of Similarity (DIBS)*. We also probe the equivalence between online optimization and offline generalization for similarity tasks. We observe that an online/offline correspondence in similarity learning can be affected by both the network architecture and label noise.

## 1. Introduction

There are key differences between similar-different discrimination and classification. For similarity learning, the relation among features is crucial but not necessarily the features themselves. In this work, we investigate how under- and

over-parameterized deep neural networks (DNNs) generalize similarity relations via two frameworks: *double descent* and *online/offline learning correspondence*, which we describe in the following.

The empirical success of overparameterized DNNs challenges conventional wisdom in classical statistical learning as it is widely known among practitioners that larger models (with more parameters) usually obtain better generalization (Szegedy et al., 2015; Huang et al., 2019; Radford et al., 2019). The double descent (Belkin et al., 2019) behavior connects “classical” and “modern” machine learning by observing that once the model complexity is large enough to interpolate the dataset (i.e., when training error reaches zero), the test error decreases again. This same pattern has been empirically demonstrated for several models and datasets, ranging from linear models (Loog et al., 2020) to modern DNNs (Spigler et al., 2019; Nakkiran et al., 2020).

Here we investigate the double descent phenomenon in similarity learning by examining in detail its behavior in Siamese networks (Bromley et al., 1994; Chopra et al., 2005). A Siamese architecture is made of two identical networks sharing weights and biases that are simultaneously updated during supervised training. The two networks are connected by a final layer, which computes the distance between branch outputs. Siamese Neural Networks (SNNs) are trained using pairs of data that are labeled as similar or different. The task of a successfully trained network is to decide if the pair samples belong to the same class. A Siamese-like setup offers a rich phenomenology to study similarity learning in the presence of noise expected in real-world data. Several subtleties should be taken into account in selecting the data and creating the pairs. In particular, we note that if the pairs are created from populations with different levels of image diversity, the resulting learning model is different, even if the total number of training pairs is fixed. Interestingly, SNNs allow for two sources of label noise in the training dataset: Single Label Noise (SLN) and Pair Label Noise (PLN) that we introduce below (see the top of Fig. 1 for an illustration).

**Single Label Noise (SLN).** Let us consider a dataset with  $N$  samples  $X^S = \{x_1, x_2, \dots, x_N\}$  belonging to  $n_c$  classes and their corresponding labels  $Y^S = \{y_1^S, y_2^S, \dots, y_N^S\}$ . If some label noise is present in the original dataset, this will propagate to the training pairs as these are created. If SLN

---

\*Equal contribution. <sup>1</sup>Work performed in part while at the International Centre for Theoretical Physics (ICTP) in Trieste and at Queen Mary University of London (QMUL). <sup>2</sup>Deutsches Elektronen-Synchrotron, DESY, Notkestraße 85, 22607 Hamburg, Germany. Correspondence to: Nayara Fonseca <nayara.focs@gmail.com>, Veronica Guidetti <veronica.guidetti@desy.de>.

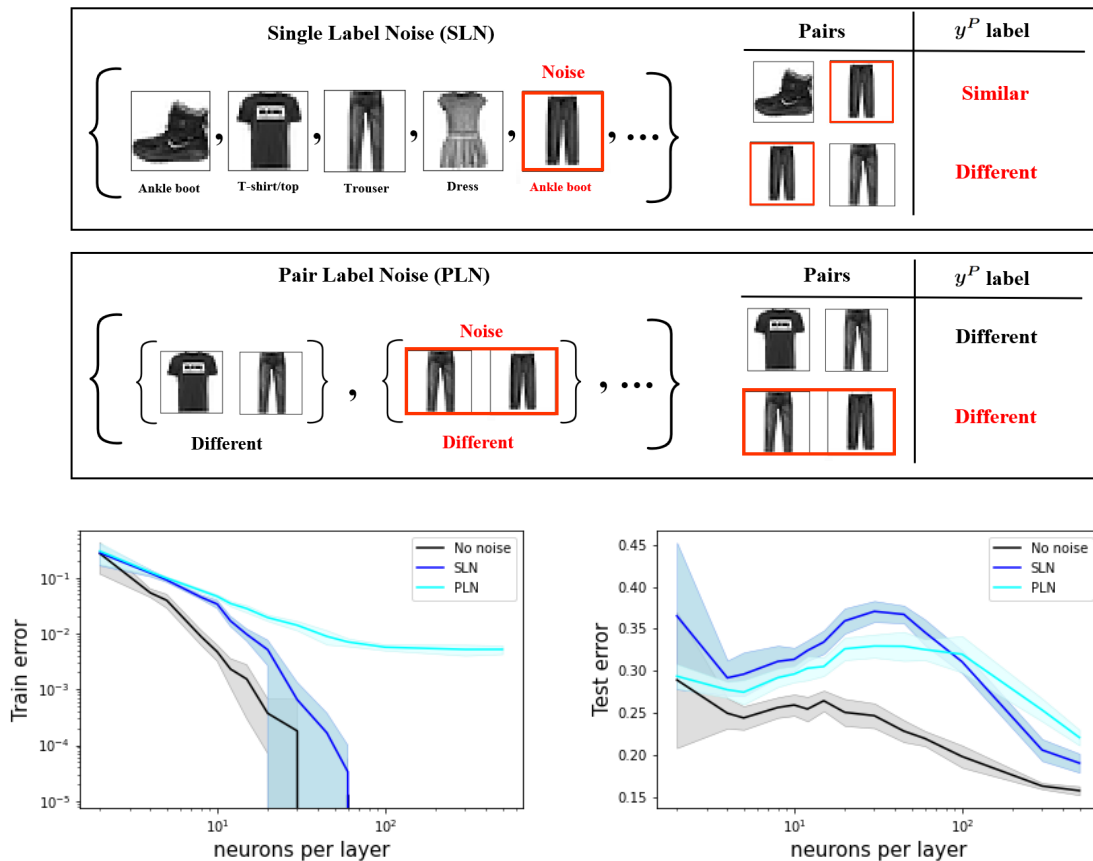


Figure 1. Top: Pictorial view about the two types of random label noise in a Siamese-like setup. Bottom: Train error (left) and test error (right) as a function of model size. We train a 3-layer MLP architecture with ReLU activation function on FMNIST with 10% label noise. We considered dense connections in the training pair data (scenario 2 in Sec. 2.1). We train the model using Adam for 2k epochs.

is uniformly introduced across all classes, it will keep the original class balancing on average (over multiple samples). On the other hand, in every single run, statistical fluctuations of uniform distribution introduce some asymmetry in the original class representative number (see left panel of Fig. 2). Finally, in SLN, similarity relations (reflexive, symmetric, and transitive properties) are preserved as mislabeling appears in all pairs containing a misclassified image.

**Pair Label Noise (PLN).** Let us now consider a dataset of  $2N$  pairs  $X^P = \{\{x_1, x_2\}, \dots, \{x_i, x_j\}, \dots, \{x_N, x_k\}\}$  with pair labels  $Y^P = \{y_1^P, y_2^P, \dots, y_{2N}^P\}$ , which can be similar ( $y^P = 1$ ) or different ( $y^P = 0$ ). Suppose we randomly shuffle some fraction of the total labels. In that case, the noise we introduce is symmetric under similar  $\leftrightarrow$  different changes, and it acts democratically on every class of the original dataset. On the other hand, PLN can lead to inconsistent relations in the pairs dataset. Indeed, as we will show in the following sections, it breaks transitivity and, therefore, similarity.

We also study similarity learning by testing the equivalence between online optimization and offline generalization. This

online/offline correspondence was proposed for supervised image classification in (Nakkiran et al., 2021), see also (Bousquet & Elisseeff, 2001; Bottou & LeCun, 2004; 2005; Hardt et al., 2016). The conjecture states that understanding generalization in an offline setting can be effectively reduced to an optimization problem in the infinite-data limit. Just as the double descent pattern described above, this framework connects under- and overparameterized limits. On the other hand, here, it is the training dataset size that dictates the two regimes. In fact, under certain conditions, online and offline test soft-errors match each other for classification tasks (Nakkiran et al., 2021). This phenomenon can be interpreted as a correspondence between overparameterized models (with a finite number of samples) and underparameterized models (with a large dataset). Here we test this idea for similarity problems.

In this work, we use both the double descent behavior and the online/offline correspondence framework to understand how networks generalize the concept of similarity. As discussed later, noisy data can impact how models learn similarity relations in distinct ways. Notably, it is empirically known that the double-descent curve is exacerbated in the

presence of random label noise in supervised classification (Nakkiran et al., 2020). Our experiments in Sec. 2.4 show that double descent appears in SNNs and is distinctly manifested for SLN and PLN scenarios when similarity relations in the training set are dense (we discuss sparse and dense connections in detail in Sec. 2.1). An example of this behavior is shown in the bottom right plot of Fig. 1. Here we trained a Siamese network made of two fully connected branches on the Fashion-MNIST (FMNIST) dataset (Xiao et al., 2017). In this case, dense pairs are formed using a reduced version of the original FMNIST. We clearly see that, for dense connections, SLN outperforms PLN in the overparametrized region. The difference between DD curves comes from the similarity-breaking nature of PLN that becomes manifest when input data are highly connected. We call this phenomenon *Density-Induced Break of Similarity (DIBS)*. This property of PLN, i.e., similarity breaking, may appear in self-supervised contrastive learning. Indeed, as exposed in (Huynh et al., 2022), if negative pairs are formed by sampling views from different images, regardless of their semantic information, this may lead to the appearance of false-negative pairs. This can break transitivity and, therefore, similarity, compromising the training efficiency. Indeed, as we show in Fig. 5, this is precisely the unbalancing effect we found in the asymptotic training error in the presence of PLN.

We test our results on MNIST and FMNIST datasets using Multi-Layer Perceptron (MLP) and Convolution Neural Network (CNN) branch architectures. To show that the DD behavior in the presence of noise only depends on the dataset quality and pairs topology, we consider two different training setups. In the first case, we compute the Euclidean distance in the output layer training the network using Contrastive Loss (Hadsell et al., 2006), in the other case, we compute the cosine similarity training the network using Cosine Embedding Loss. All the plots related to the different datasets, architectures, sparse and dense similarity connections, and training setups can be found in App. D.1.

This work investigates how neural networks learn similarity relations and how the learning model is affected by noise. Our main contributions are:

- We point out that similarity learning may be affected by two distinct noise sources and study the double-descent behavior in Siamese-like architectures. Our findings indicate that, for the same amount of effective noise, SLN outperforms PLN on densely connected datasets in the overparametrized regime.
- We identify a phenomenon we call *Density-Induced Break of Similarity (DIBS)*. We find that PLN transitivity breaking introduces significant inconsistencies into the training labels of dense datasets. We show that the interpolation threshold (training error = 0) cannot be achieved in this scenario, and we derive the analytic

formula for the asymptotic training error value in the deep overparametrized regime.

- We test the correspondence between offline generalization and online optimization for similarity learning. We discuss how the architecture and the presence of noisy labels can impact differently on these two regimes. In particular, the effect of label noise is more important in the offline case.

### 1.1. Related Work

Among all possible similarity learning methods, contrastive learning (Chopra et al., 2005; Hadsell et al., 2006; Oord et al., 2018) has become one of the most prominent supervised (Khosla et al., 2020; Gunel et al., 2020) and self-supervised (Bachman et al., 2019; Tian et al., 2020a; He et al., 2020; Chen et al., 2020) ML techniques to learn representations of high-dimensional data, producing impressive results in several fields (Le-Khac et al., 2020; Jaiswal et al., 2021). Despite its success, contrastive learning usually requires huge datasets often created using data augmentation techniques. In supervised learning tasks, this compensates for the lack of infinite amounts of human-labeled data. On the other hand, real-life data are noisy. Therefore, a major challenge in further improving the accuracy and efficiency in similarity learning is to understand how NNs react to different noise sources and develop robust models, generalizing on real and, therefore, noisy datasets. Although there are several works investigating classification tasks in deep neural networks in the presence of noise (e.g., (Li et al., 2019; Han et al., 2019; Arazo et al., 2019; Harutyunyan et al., 2020; Song et al., 2020)), much less attention was given to contrastive learning with noisy data. In fact, only very recently, some attention was devoted to find the impact of noisy data augmentation and to develop robust contrastive learning setups (Tian et al., 2020b; Miech et al., 2020; Morgado et al., 2021; Chuang et al., 2022).

## 2. Similarity Learning

Let us start this section by defining the criteria we used to construct the pairs dataset. Indeed, this is an arbitrary choice that may considerably impact the final result. As opposed to classification problems, where the main concerns during dataset creation are class balancing and image diversity, in contrastive learning, we should consider that relations among pairs (or groups) of images define an unoriented graph of similarity relations inside the input space.

Calling  $N$  the total number of images in the full dataset, the density of this graph,  $\rho = |N_{\text{pairs}}| / \binom{N}{2}$ , tells us how much information we have about the input images. Therefore, to maximize the information about a certain dataset, we should construct all possible pairs,  $\binom{N}{2} \sim N^2$ , but this quickly becomes unfeasible when considering large datasets. For this reason, we construct pairs in a way that maximizes the

information about similar images (all similar images are transitivity-related) and scales linearly with  $N$ . In practice, we construct closed chains of positive pairs within the same class,  $c$ ,  $\{\{x_1^c, x_2^c\}, \dots, \{x_k^c, x_k^c\}, \dots, \{x_n^c, x_1^c\}\}$ , where  $n$  is the total number of images in  $c$ . Then, to build negative pairs, each image is connected to a randomly chosen one belonging to a different class. If the original dataset classes are balanced, each image appears on average in 4 different pairs (2 times in the positive and 2 times in the negative pairs). Therefore, the total number of pairs is given by  $N_{\text{pairs}} = 2 \times N = 2 \times n \times n_c$ , where  $n_c$  is the total number of classes.<sup>1</sup>

The following sections describe the procedure to form the pairs, our experimental setup, and how noise is introduced in the training data.

### 2.1. Sparse and Dense Connections

The topology of the dataset crucially affects the resulting learning model. Here, we provide details regarding the two methods we use to form pairs in our experiments. These lead to setups with sparse and dense connections in the input space. In the left panel of Fig. 2, we show a pictorial view of the data relations in these two scenarios in the absence of noise, and the SLN and PLN cases. Below, we describe how PLN and SLN are added to each case.

**Scenario 1: Sparse connections.** To train the network in the absence of noise, we first create the pairs using the full dataset. We follow the procedure described at the beginning of this section so that  $N_{\text{pairs}} = 2 \times N$ . We then take  $N_{\text{sample}}$  balanced pairs from the  $N_{\text{pairs}}$  list to train the NN and repeat this procedure  $n_s$  times. Below, we describe the strategy used to introduce PLN and SLN in this scenario.

---

#### *Sparse* Pair Label Noise (PLN)

---

```

load(dataset)
CREATEBALANCEDPAIRS(dataset)
for  $i = 1, 2, \dots, n_s$  do
    fraction of  $N_{\text{total}}$  labels  $\leftarrow$  random(0,1)
    select balanced  $N_{\text{sample}}$  pairs from  $N_{\text{pairs}}$ 
    train
end for
    
```

---

**Scenario 2: Dense connections.** In this setup, we create a reduced version of the original dataset. Being interested in training the network on  $N_{\text{pairs}}$  pairs, we select  $N_{\text{reduced}} = N_{\text{pairs}}/2$  images from the original training set. The reduced dataset is balanced so that we have  $N_{\text{pairs}}/(2n_c)$  images per class. Then, we create our training and test samples using the same prescription described at the beginning of this section. We connect adjacent images within the same

<sup>1</sup>Note that this formula holds for dataset with at least 2 images per class.

---

#### *Sparse* Single Label Noise

---

```

load(dataset)
for  $i = 1, 2, \dots, n_s$  do
    fraction of  $N$  labels  $\leftarrow$  random(1, $n_c$ )
    dataset  $\leftarrow$  labels with noise
    CREATEBALANCEDPAIRS(dataset)
    select balanced  $N_{\text{sample}}$  pairs from  $N_{\text{pairs}}$ 
    train
end for
    
```

---

class and each of them with a random image belonging to a different class so that we get exactly  $N_{\text{pairs}}$  pairs that will be automatically balanced. We repeat this procedure  $n_s$  times. The setup used to introduce PLN and SLN in this scenario is described below.

---

#### *Dense* Pair Label Noise

---

```

load(dataset)
for  $i = 1, 2, \dots, n_s$  do
    rdataset  $\leftarrow$  REDUCEDDATASET(dataset)
    CREATEBALANCEDPAIRS(rdataset)
    fraction of  $N_{\text{reduced}}$  labels  $\leftarrow$  random(0,1)
    train
end for
    
```

---



---

#### *Dense* Single Label Noise

---

```

load(dataset)
for  $i = 1, 2, \dots, n_s$  do
    rdataset  $\leftarrow$  REDUCEDDATASET(dataset)
    fraction of  $N$  labels  $\leftarrow$  random(1, $n_c$ )
    rdataset  $\leftarrow$  labels with noise
    CREATEBALANCEDPAIRS(rdataset)
    train
end for
    
```

---

The functions CREATEBALANCEDPAIRS and REDUCEDDATASET describing the exact steps to prepare the dataset are given in App. A.

### 2.2. Effective noise

To make a consistent comparison, we need to introduce the same amount of input label noise in both the SLN and the PLN cases. Being  $n_c$  the number of image classes,  $y_i^S$  the label of the  $i$ -th image, and  $y_i^P$  the label of the  $i$ -th pair of images, we can define the SLN transformation that is applied to the whole dataset as

$$\mathcal{T}(q) : y_i^S \rightarrow \text{Rnd}(0, n_c - 1) \quad \text{with probability } q \quad (1)$$

and the PLN transformation as

$$\mathcal{T}_{\mathcal{P}}(\tilde{q}) : y_i^P \rightarrow \text{Rnd}(0, 1) \quad \text{with probability } \tilde{q}. \quad (2)$$

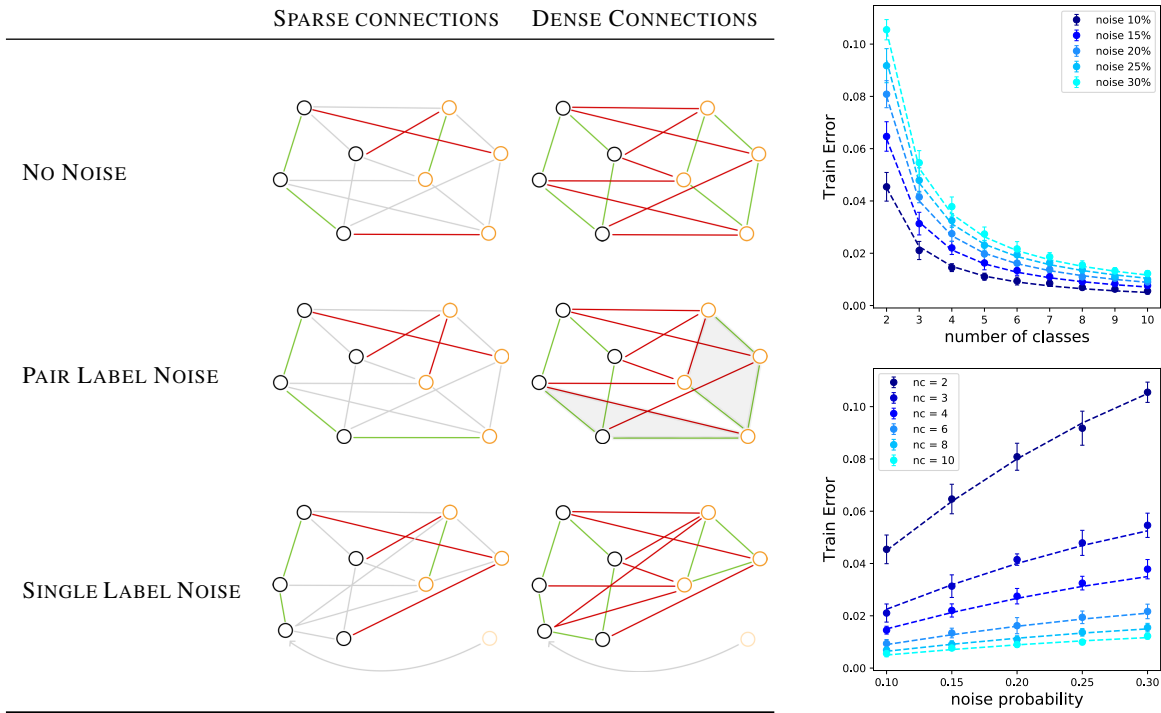


Figure 2. Left: Pictorial view of data relation appearing in Scenario 1 (left) and 2 (right) for two classes of data (represented by black and orange dots). Positive pairs are connected by green edges, negative pairs by red edges. Scenario 1 implies sparse connections among data (ignored connections and data are in light gray), while Scenario 2 is densely connected. Gray shaded areas are examples of DIBS in the presence of PLN. Right: Comparison between analytic and numerical estimates of the asymptotic training error behavior at varying number of classes  $n_c$  (top) and noise (bottom) in the presence of PLN in scenario 2. We consider the FMNIST dataset and compute the value of the training error coming from a 3-layers MLP architecture with 500 neurons per layer after 4k epochs. The Siamese output layer computes the Euclidean distance, and we train the network using contrastive loss. Numerical results (bullet points with standard error bars) come from 10 runs where we choose different random classes each time. Analytic estimates (dashed lines) come from Eq. (14).

As the introduction of SLN happens before pair creation and the pairs are constructed so that the dataset is balanced, i.e., half pairs are equal, and half are different, the probability of effective pair mislabeling induced by SLN,  $P_{\text{SLN}}(q)$ , is given by

$$P_{\text{SLN}}(q) = q - \frac{q^2}{2}. \quad (3)$$

while the probability of effective pair mislabeling coming from PLN,  $P_{\text{PLN}}(\tilde{q})$ , is

$$P_{\text{PLN}}(\tilde{q}) = \frac{\tilde{q}}{2}. \quad (4)$$

The requirement of having the same amount of noise in the dataset ( $P_{\text{SLN}}(q) = P_{\text{PLN}}(\tilde{q})$ ) boils down to the following relation between  $q$  and  $\tilde{q}$ :

$$q = 1 - \sqrt{1 - \tilde{q}}. \quad (5)$$

The full derivation of these results is given in Appendix B.

We want to stress that PLN and SLN impose different constraints on the training process. PLN is a balanced noise

source as the probability of transforming even pairs into odd ones, and vice versa is the same. On the other hand, SLN is an unbalanced source of noise, i.e., the probability that Eq. (1) transforms equal pairs into different ones,  $(n_c - 1)/n_c$ , is in general much higher than the opposite case,  $1/n_c$ . Moreover, as opposed to classification tasks, in Siamese networks and contrastive learning, noise can generally lead to inconsistent relations in the training set. A similarity relation is defined by reflexivity, symmetry, and transitivity, but the appearance of noise can compromise this last property. In fact, PLN, randomly shuffling pair labels, leads to inconsistent relations in the dataset (see Fig. 2). This effect becomes more apparent as we increase the density (number of links in Fig. 2) in our training set. On the other hand, similarity breaking does not appear in SLN, where the similarity relations may go against image features but are always self-consistent.

### 2.3. Output layer and Loss function

We perform our experiments using the two different output layers and loss functions described below.

**Euclidean distance and Contrastive Loss.** In this first case, the Siamese NN output layer computes the Euclidean distance between the output vectors of the Siamese branches,  $\bar{z}(x)$ . Therefore, the model prediction that quantifies the similarity between the two images in a pair is given by:

$$d_i = \|\bar{z}(x_1^i) - \bar{z}(x_2^i)\|_2. \quad (6)$$

We then train the network considering the contrastive loss function (Hadsell et al., 2006):

$$\mathcal{L}(y^P, d) = \frac{1}{N_{\text{pairs}}} \sum_i \left[ y_i^P d_i^2 + (1 - y_i^P) [\max(0, m - d_i)]^2 \right], \quad (7)$$

where  $y_i^P$  is the true label and  $m$  sets the threshold at which the network classifies a given pair as similar or different. During training, the network tries to minimize  $\mathcal{L}$  by collapsing similar samples and pulling apart different samples by a distance equal to the margin,  $m$ . The accuracy is given by:

$$\text{acc} = 1 - \text{err} = 1 - \frac{1}{2N} \sum_i |y_i^P - \hat{y}(d_i)|, \quad (8)$$

$$\text{where } \hat{y}(d_i) = [\mathbb{1}_{d < m/2}](d_i). \quad (9)$$

In all experiments, we choose the margin to be  $m = 1$ .

**Cosine similarity and Cosine Embedding Loss.** In this setup, the output layer computes the cosine similarity between the output vectors of the Siamese branches. The model prediction is thus given by:

$$s_i = \cos \left( \frac{\bar{z}(x_1^i) \cdot \bar{z}(x_2^i)}{\|\bar{z}(x_1^i)\|_2 \|\bar{z}(x_2^i)\|_2} \right). \quad (10)$$

We train the network using the Cosine Embedding Loss function,

$$\mathcal{L}(y^P, s) = \frac{1}{N_{\text{pairs}}} \sum_i \left[ y_i^P (1 - s_i) + (1 - y_i^P) \max(0, s_i - \cos(\alpha)) \right], \quad (11)$$

according to which similar images should give rise to vectors pointing in roughly the same direction. In contrast, the angle between vectors coming from different images should be larger than or equal to  $\alpha$ . In this model, we compute the accuracy as:

$$\text{acc} = 1 - \frac{1}{2N} \sum_i |y_i^P - \hat{y}(s_i)|, \quad (12)$$

$$\text{where } \hat{y}(s_i) = [\mathbb{1}_{s > \cos(\alpha/2)}](s_i). \quad (13)$$

In all experiments, we choose  $\alpha = \pi/3$ .

## 2.4. Architecture and Optimizer

In this work, we consider two simple Siamese branch architectures. The first one is an MLP with 3 hidden layers having the same width and ReLU activation functions with Xavier uniform initialization (Glorot & Bengio, 2010). The second architecture is a 4-layer CNN based on the model described in (Page, 2018). It contains three Convolution-BatchNormalization-ReLU-MaxPooling layers and a fully-connected output layer. The number of filters in each convolution layer scales as  $[k, 2k, 2k]$  while the MaxPooling is  $[1, 2, 8]$ . We fix the kernel size = 3, stride = 1 and padding = 1. When we train the network using contrastive loss (cosine embedding loss), we set the fully-connected output layer width to  $k$  ( $2k$ ). To understand the impact of overparameterization, we study how training and test errors vary at increasing network width. Namely, we increase the number of neurons per layer in the fully connected architecture and the parameter  $k$  in the CNN.

In every DD experiment, we let the network evolve for 2000 epochs using Adam optimizer with minibatches of size 128 and learning rate  $\lambda = 10^{-4}$ . All our experiments make use of the TensorFlow/Keras framework (Abadi et al., 2015).

## 3. Results

### 3.1. Double Descent in Similarity Learning

We study double descent on MNIST and FMNIST datasets in scenarios 1 (Sparse) and 2 (Dense). For all datasets, we consider 6000 training pairs and 9000 test pairs. We run 15 (10) evolutions of the MLP (CNN) network using different training and test samples at each time. Our results, showing the average training and test errors together with error bars, can be found in Fig. 1 and App. D.1. In all examples, we can see the double descent phenomenon, regardless of the architecture, the loss function, the scenario, and the noise. Nevertheless, as expected, DD becomes more prominent in the presence of noise. In all the experiments discussed, we considered  $\tilde{q} = 0.2$ , i.e., an effective noise of 10%. In Fig. 8, we show how the network reacts to different amounts of noisy labels.

In Scenario 1, the input dataset connections are sparse, and PLN and SLN have the same impact on training. This makes sense as there should not be any difference between PLN and SLN in the extreme case where every image appears only once in the training set. A closer look at the plots related to scenario 1 shows that SLN test error tends to be slightly higher than PLN one. This happens because SLN is a slightly unbalanced noise source. Indeed, we experimentally saw that unbalanced noise generically leads to higher test errors.

In Scenario 2, input connections are dense, and the system behaves differently under SLN and PLN. SLN test error



tends to be higher from small to medium network sizes. A hint about how this happens is given in Fig. (2). Indeed, on top of being asymmetric, SLN introduces a systematic error: a mislabeled image appears to be mislabeled in every pair. Therefore, given that the image features are not going to agree with pair labels, the only way the network has to classify correctly is by extracting the image from its natural distribution. Being NNs continuous functions, this implies that a neighborhood of said image must be extracted as well, increasing the test error. At higher network widths, the volume of the mislabeled image neighborhood can become arbitrarily small, and the test error is free to go down again. On the other hand, PLN stays higher in the deep overparametrized regime. Indeed, randomly changing similarity relations in the input dataset, PLN ends up breaking transitivity, thus making the training labels inconsistent. Beyond keeping test error high, this inconsistency also implies that the network is not able to overfit the training data completely: the training error will no longer vanish just by increasing the number of network parameters and reaches an asymptotic value given by:

$$\lim_{n_\theta \rightarrow \infty} \text{TrainError}_{\text{Dense}}^{\text{PLN}}(P, n_c) = \frac{P(1-P)}{2(n_c-1)} \quad (14)$$

where  $n_\theta$  is the number of network parameters and  $P = P_{\text{PLN}}(\tilde{q})$  is the effective amount of noise. In Fig 2 we validate our formula by comparing it with experimental results showing how, in the overparametrized regime, the training error changes with the effective noise and with the different number of classes.

### 3.2. Similarity Generalization: online vs. offline

In this section, we probe the Deep Bootstrap framework (Nakkiran et al., 2021) in the context of similarity learning. This framework introduces a correspondence between online and offline settings for classification problems. The proposal establishes a relation between the *Real World* (offline), which has a fixed number of samples in the training set, with the *Ideal World* (online), where the optimizer’s updates always use fresh samples. The relation between Real and Ideal scenarios is manifested by the following alternative decomposition of the test error (Nakkiran et al., 2021):

$$\begin{aligned} \text{TestError}(f_t) &= \text{TestError}(f_t^{\text{iid}}) \\ &+ [\text{TestError}(f_t) - \text{TestError}(f_t^{\text{iid}})], \end{aligned} \quad (15)$$

where,  $f_t$  refers to a neural network after  $t$  optimizer steps in the Real World with a fixed number of samples, and  $f_t^{\text{iid}}$  is a network in the Ideal World that is trained as  $f_t$  but without re-using samples. This means that the optimization over minibatches gets new samples at each gradient iteration in the online setting. Note that the decomposition in Eq. (15) makes the *bootstrap error* (the subtraction in brack-

ets) explicit.<sup>2</sup> This measures how well models trained offline represent the online regime. The first term in the right-hand side of Eq. (15) is the test error in the Ideal World, which quantifies how fast the network optimizes on the population loss. Nakkiran et al. (2021) give empirical evidence that an online/offline correspondence holds for classification problems by showing that the bootstrap error is small in several network models for supervised image classification. The conjecture suggests that understanding generalization in the Real World can be effectively reduced to an optimization problem in online learning. Here, we test if this correspondence is also valid for similarity tasks.

We use the digit section of EMNIST (Cohen et al., 2017) that is an extended version of the standard MNIST dataset, with 240,000 training (and 40,000 test)  $28 \times 28$  greyscale pixel images. The networks are trained with the contrastive loss function in Eq. (7). We train Real World over 40 epochs using 12k pairs that are created considering Sparse and Dense scenarios, as described in Sec. 2.1. The Ideal World is trained once on 480k pairs created using the full training set of 240k samples. We test the models with 9k pairs constructed from the test set and consider Siamese networks with MLP and CNN blocks. The architecture details are given in Sec. 2.4.

Real- and Ideal-world scenarios in the absence of label noise are shown in Fig. 3, where offline and online settings are compared after the same number of training iterations. We consider the Siamese architectures described in Sec. 2.4, using 200 nodes per layer for the MLP cases (total of 237,400 parameters) and width  $k = 47$  (total of 235,611 parameters) for the CNN cases. In Fig. 3, we plot the median over 5 trials (random network initialization) for the Real-world cases with MLP blocks, and the median over 7 trials for all CNN cases. The final training errors for the MLP (CNN) Real world scenarios are below 1% (3%), see Fig. 17 in App. D.2.

Figure 3 (left) shows that the MLP Real test error for dense connections is slightly higher than the sparse one, and there is a constant gap between the two scenarios, which converges after about 2k iterations. Both curves present a small deviation from the Ideal World, which marginally increases as the online test error improves with more fresh samples. Interestingly, the Siamese network with CNN blocks in the Ideal World is close to the MLP Real case for dense connections. On the other hand, there is a larger bootstrap gap between the CNN Real scenarios and the corresponding Ideal case. In addition, CNN offline test error curves start to perform worse (at about 900 iterations), while test error improves in the ideal regime. This signals that the Real test errors for the SNN with CNN blocks overfits earlier

<sup>2</sup>See Appendix C in (Nakkiran et al., 2021) for connections to the nonparametric bootstrap in statistics (Efron, 1979; Efron & Tibshirani, 1986).

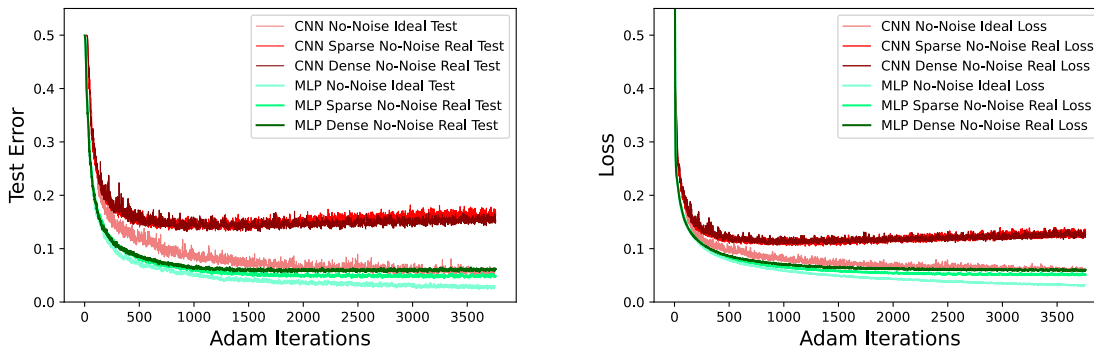


Figure 3. Ideal vs. Dense/Sparse Real worlds in the absence of label noise. Plots show Test Errors (left) and Test Losses (right) as a function of minibatch Adam iterations for the MLP (nodes per layer = 200) and CNN ( $k = 47$ ) Siamese architectures described in Sec. 2.4.

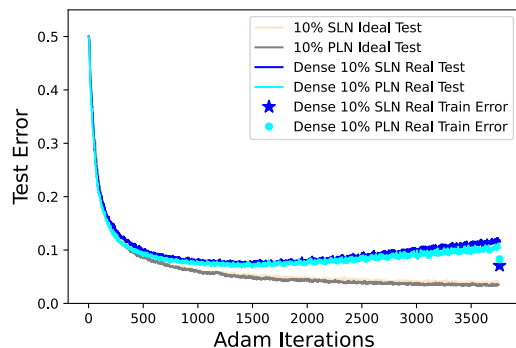


Figure 4. Ideal vs. Dense Real worlds with 10% of label noise for the MLP architecture with 200 nodes per layer. The plots shows the Test Errors as a function of minibatch Adam iterations. The star (dot) corresponds to the SLN (PLN) Real World Train Error at the end of training.

than the one with MLP blocks. This can be avoided by stopping training earlier in the CNN case. However, since we want to compare the architectures, we train both MLP and CNN scenarios for the same number of optimizer iterations. We stress though, that (Nakkiran et al., 2021) compare soft-errors, instead of the hard-errors that we use here.<sup>3</sup> In the classification examples discussed in (Nakkiran et al., 2021), they found that the bootstrap gap is often smaller for soft-errors. Besides the test errors, it is interesting to compare other metrics between online/offline settings. In particular, note that the Real/Ideal Test Losses (right-panel of Fig. 3) follow the same behavior as their corresponding Test Errors.

We also study the impact of noisy labels on Real and Ideal scenarios. In Fig. 4, we consider 10% of label noise for the MLP architecture and plot the median over 5 trials (random network initialization and random noise sampling selection).

<sup>3</sup>The soft-error is given by soft-error = 1 - soft-accuracy, where soft-accuracy is the softmax probability on the right label.

We note that both Ideal and Real test errors are affected by noise, but this effect is exacerbated in the Real World. This can be understood because “fresh” samples add more diversity into the model, which improves generalization even if these new samples present noisy labels. In Fig. 4, the Dense-Real cases start to overfit at about 1k minibatch updates, while the Ideal World continues to improve generalization. We observe a similar behavior for the sparse case and the losses in Fig. 16 (App. D.2). Notice that the test errors in Fig. 4 should not be directly compared to the double-descent curves in Sec. 3.1, where there is a clear difference between PLN and SLN for the scenario with dense connections. While in Sec. 3.1 the network is trained over 2k epochs, such that training continues after the training error has reached its asymptotic value; here we want to probe the Real World when its optimizer is still updating significantly. As pointed out in previous works (see, e.g., (Nakkiran et al., 2020)), setups with early stopping training usually do not exhibit double descent. For the Real-world models in this section, the training always stops at 40 epochs.

## 4. Conclusions

This work studies generalization in similarity learning with noisy labeled data focusing, in particular, on SNNs. We find that DD clearly appears in SNNs and becomes more evident in the presence of noisy labels. We check DD using different architectures, datasets, and loss functions. We present two kinds of noise, SLN, and PLN, that may affect the input data. While SLN preserves similarity relations, PLN breaks transitivity. We analytically prove and experimentally show that similarity-breaking noise sources deeply affect generalization performances. The same noise sources presented in this work can be easily generalized to models where the network input is given by multiple images. We also investigate the equivalence between online optimization (infinite-data regime) and offline generalization (finite number of samples) for similarity problems. Our results

indicate that both the network architecture and the existence of noisy labels can disturb an online/offline correspondence for similarity tasks.

## 5. Acknowledgement

This work was partially supported by the *CoSubmitting Summer (CSS) program at ICLR 2022*. This research was supported in part through the *Maxwell* computational resources operated at *Deutsches Elektronen-Synchrotron DESY*, Hamburg, Germany. We thank Preetum Nakkiran for useful discussions and for proposing the idea of probing the online/offline correspondence in Siamese networks. We also thank Alexander Westphal, Gonalo Valadao and Arun Raja for useful discussions.

## References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- Arazo, E., Ortego, D., Albert, P., O’Connor, N. E., and McGuinness, K. Unsupervised label noise modeling and loss correction. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 312–321. PMLR, 2019. URL <http://proceedings.mlr.press/v97/arazo19a.html>.
- Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.
- Belkin, M., Hsu, D., Ma, S., , and Mandal, S. Reconciling modern machine-learning practice and the classical bias-variance trade-off. In *Proceedings of the National Academy of Sciences of the United States of America vol. 116,32 (2019): 15849-15854*, 2019. URL <https://doi.org/10.1073/pnas.1903070116>.
- Bottou, L. and LeCun, Y. Large scale online learning. In Thrun, S., Saul, L., and Scholkopf, B. (eds.), *Advances in Neural Information Processing Systems 16 (NIPS 2003)*. MIT Press, Cambridge, MA, 2004. URL <http://leon.bottou.org/papers/bottou-lecun-2004>.
- Bottou, L. and LeCun, Y. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2):137–151, 2005. URL <http://leon.bottou.org/papers/bottou-lecun-2004a>.
- Bousquet, O. and Elisseeff, A. Algorithmic stability and generalization performance. *Advances in Neural Information Processing Systems*, pp. 196–202, 2001.
- Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., and Shah, R. Signature verification using a ”siamese” time delay neural network. In Cowan, J., Tesauro, G., and Alspector, J. (eds.), *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1994. URL <https://proceedings.neurips.cc/paper/1993/file/288cc0ff022877bd3df94bc9360b9c5d-Paper.pdf>.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.
- Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, volume 1, pp. 539–546 vol. 1, 2005. doi: 10.1109/CVPR.2005.202.
- Chuang, C.-Y., Hjelm, R. D., Wang, X., Vineet, V., Joshi, N., Torralba, A., Jegelka, S., and Song, Y. Robust contrastive learning against noisy views. *arXiv preprint arXiv:2201.04309*, 2022.
- Cohen, G., Afshar, S., Tapson, J., and van Schaikf, A. EMNIST: an extension of MNIST to handwritten letters. 2017. URL <https://github.com/hosford42/EMNIST>.
- Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. ISSN 00905364. URL <http://www.jstor.org/stable/2958830>.
- Efron, B. and Tibshirani, R. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical science*, pp. 54–75, 1986.
- Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

- Gunel, B., Du, J., Conneau, A., and Stoyanov, V. Supervised contrastive learning for pre-trained language model fine-tuning. *arXiv preprint arXiv:2011.01403*, 2020.
- Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742, 2006. doi: 10.1109/CVPR.2006.100.
- Han, J., Luo, P., and Wang, X. Deep self-learning from noisy labels. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 5137–5146. IEEE, 2019. doi: 10.1109/ICCV.2019.00524. URL <https://doi.org/10.1109/ICCV.2019.00524>.
- Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pp. 1225–1234. PMLR, 2016.
- Harutyunyan, H., Reing, K., Steeg, G. V., and Galstyan, A. Improving generalization by controlling label-noise information in neural network weights. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4071–4081. PMLR, 2020. URL <http://proceedings.mlr.press/v119/harutyunyan20a.html>.
- He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.
- Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32:103–112, 2019.
- Huynh, T., Kornblith, S., Walter, M. R., Maire, M., and Khademi, M. Boosting contrastive self-supervised learning with false negative cancellation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2785–2795, 2022.
- Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2021.
- Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.
- Le-Khac, P. H., Healy, G., and Smeaton, A. F. Contrastive representation learning: A framework and review. *IEEE Access*, 2020.
- Li, J., Wong, Y., Zhao, Q., and Kankanhalli, M. S. Learning to learn from noisy labeled data. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 5051–5059. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00519. URL [http://openaccess.thecvf.com/content\\_CVPR\\_2019/html/Li\\_Learning\\_to\\_Learn\\_From\\_Noisy\\_Labeled\\_Data\\_CVPR\\_2019\\_paper.html](http://openaccess.thecvf.com/content_CVPR_2019/html/Li_Learning_to_Learn_From_Noisy_Labeled_Data_CVPR_2019_paper.html).
- Loog, M., Viering, T., Mey, A., Krijthe, J. H., and Tax, D. M. A brief prehistory of double descent. *Proceedings of the National Academy of Sciences*, 117(20):10625–10626, 2020.
- Miech, A., Alayrac, J.-B., Smaira, L., Laptev, I., Sivic, J., and Zisserman, A. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9879–9889, 2020.
- Morgado, P., Misra, I., and Vasconcelos, N. Robust audio-visual instance discrimination. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12934–12945, 2021.
- Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=B1g5sA4twr>.
- Nakkiran, P., Neyshabur, B., and Sedghi, H. The deep bootstrap framework: Good online learners are good offline generalizers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL <https://openreview.net/forum?id=guetrIHLEFI>.
- Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- Page, D. How to train your resnet. 2018. URL <https://myrtle.ai/how-to-train-your-resnet-4-architecture/>.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

- Song, H., Kim, M., Park, D., and Lee, J. Learning from noisy labels with deep neural networks: A survey. *CoRR*, abs/2007.08199, 2020. URL <https://arxiv.org/abs/2007.08199>.
- Spigler, S., Geiger, M., d’Ascoli, S., Sagun, L., Biroli, G., and Wyart, M. A jamming transition from under-to over-parametrization affects generalization in deep learning. *Journal of Physics A: Mathematical and Theoretical*, 52 (47):474001, 2019.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
- Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 776–794. Springer, 2020a.
- Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? *arXiv preprint arXiv:2005.10243*, 2020b.
- Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. 2017. URL <https://github.com/zalandoresearch/fashion-mnist>.

## A. Pseudocode to create the balanced pairs and reduced dataset

---

### Creating Balanced Pairs

---

```

1: function CREATEPAIRS(images, labels, numClasses)
2:    $n \leftarrow$  number of samples for each class
3:   for  $d = 1, 2, \dots, \text{numClasses}$  do
4:     for  $i = 1, 2, \dots, n$  do
5:       pairs_pos = {images(i), images(i+1)}
6:        $y = (d + \text{random-integer}(1,9)) \% 10$ 
7:       pair_neg = {images(i), images(y)}
8:       pairs = pairs + {pair_pos, pair_neg}
9:       p_labels = p_labels + {1, 0}
10:    end for
11:  end for
12:  return pairs, p_labels
13: end function

```

---

### Create Reduced dataset

---

```

1: function REDUCEDATASET(images, labels, numClasses, NewSize)
2:    $n \leftarrow$  NewSize/numClasses
3:   indices=[]
4:   for  $d = 1, 2, \dots, \text{numClasses}$  do
5:     class_indices = where(labels==d)
6:     n_class_indices=random.choice(class_indices, n)
7:     indices.append(n_class_indices)
8:   end for
9:   return images[indices], labels[indices]
10: end function

```

---

## B. Effective noise derivation

We start by considering the amount of effective noise (real mislabeling) introduced by the pair label transformation

$$\mathcal{T}_P(\tilde{q}) : y_i^P \rightarrow \text{Rnd}(0, 1) \quad \text{with probability } \tilde{q}. \quad (16)$$

Each time we apply this transformation, the probability of a change in the pair label is  $1/2$ , so the effective error probability is:

$$P_{\text{PLN}} = \frac{\tilde{q}}{2}. \quad (17)$$

This computation is slightly more complicated in the SLN case. Indeed, if we apply the following transformation

$$\mathcal{T}(q) : y_i^S \rightarrow \text{Rnd}(0, n_c - 1) \quad \text{with probability } q, \quad (18)$$

on the initial dataset labels  $y_i^S$ , and we then create the pairs, the probability that one (and only one) element in a pair has been operated by  $\mathcal{T}$  is

$$P_{1L} = 2q(1 - q), \quad (19)$$

while the probability that both elements have been operated by  $\mathcal{T}$  is

$$P_{2L} = q^2. \quad (20)$$

Now the question is: what is the probability that this single label operation (we recall that the term *single label* regards the application of  $\mathcal{T}$  on the label of one or both pair elements and not on the *pair label*) leads to effective pair label corruption? Let us assume that we have a pair of images belonging to different classes  $y^P = 0$ . The probability that the transformation

of a single image label changes the pair label is equal to the likelihood that the same operation over both images effectively changes the pair label. The value of that probability is the following:

$$Q^{1 \rightarrow 0}_{1L} = Q^{1 \rightarrow 0}_{2L} = \frac{1}{n_c}. \quad (21)$$

The same reasoning can be applied to pairs of objects belonging to the same class,  $y^P = 1$ , and leads to

$$Q^{0 \rightarrow 1}_{1L} = Q^{0 \rightarrow 1}_{2L} = \frac{(n_c - 1)}{n_c}. \quad (22)$$

Creating a balanced dataset where half of the pairs are equal and half are different is common practice. Therefore, we create a dataset where

$$P_{y^P=1} = P_{y^P=0} = \frac{1}{2}. \quad (23)$$

Finally, we are now ready to estimate the amount of real noise that is introduced in our dataset corrupting single images labels. This is given by:

$$\begin{aligned} P_{\text{SLN}} &= P_{y^P=1} (P_{1L} Q^{0 \rightarrow 1}_{1L} + P_{2L} Q^{0 \rightarrow 1}_{2L}) \\ &\quad + P_{y^P=0} (P_{1L} Q^{1 \rightarrow 0}_{1L} + P_{2L} Q^{1 \rightarrow 0}_{2L}) \\ &= \frac{1}{2} (P_{1L} + P_{2L}) (Q^{0 \rightarrow 1}_{1L} + Q^{1 \rightarrow 0}_{1L}) \\ &= q - \frac{1}{2} q^2. \end{aligned} \quad (24)$$

Requiring that the effective dataset noise is the same in SLN and PLN setups, leads us to Eq. (5).

### C. Unbalanced train error with PLN DIBS

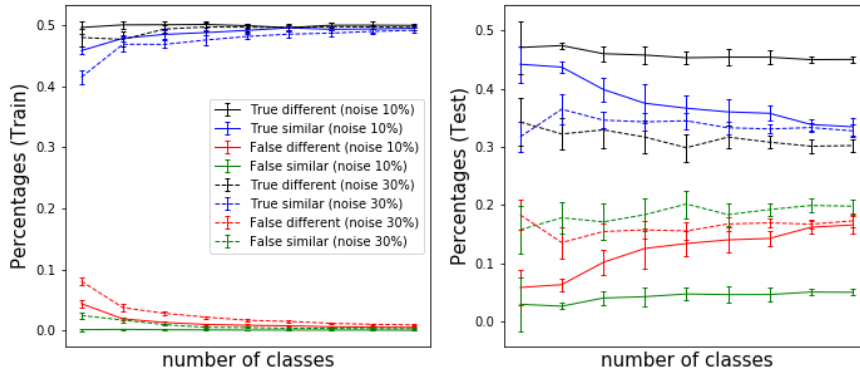


Figure 5. Results about normalized confusion matrix computed on the asymptotic training error (left) and its related test error (right) at varying number of classes and PLN. These results are referred to FMNIST using the dense pair regime (scenario 2). The experimental setup is the same described in Fig. 2.

Here, we provide some additional information about the properties of the asymptotic training error arising with PLN on a dense pairs dataset. As mentioned in the main text, PLN breaks transitivity, leading to inconsistent relations that the NN cannot satisfy. In fact, the margin-based loss function tends to collapse images belonging to the same class to a single point and separate different images by a distance equal to or greater than the margin. Some inconsistencies appearing in this setup are shown in Fig. 2. Nevertheless, it is not clear how the network deals with such contradictions at this point. The final training error could arise from three distinct behaviors: collapsing different pairs, separating equal pairs, or a mixture of the two. Contrary to the third option, the first two cases would lead to an unbalanced asymptotic train error (similar pairs systematically mislabeling or vice-versa). In order to answer this question, we compute the confusion matrix on the

asymptotic training set, varying the number of classes and PLN. This corresponds to studying the source of the results shown in the top-right corner of Fig. 2. Our findings are shown in Fig. 5. We see that the network is biased towards “False different” classification. That is, the NN tends to classify equal pairs ( $y^S = 1$ ) as different ones ( $y^S = 0$ ). We can provide some intuition about why this happens. As shown in Fig. 6 we can have transitivity breaking inside similar pairs chains and among vertices belonging to different classes. We need to notice that more than one mislabeling inside equal images chains does not lead to inconsistent classification. This implies that the impact of this mislabeling on the training error is almost negligible. On the other hand, inconsistencies on paths involving different classes always lead to inconsistencies. In this specific case, the configuration that minimizes the amount of training error is the one where the noise-induced similar pair is misclassified, leading to a “False different” kind of error. We also studied the impact of the training error unbalancing on the test error (RHS of Fig. 5). Surprisingly, while in the low noise regime (10%), the test error is also biased towards the “False different” pair classification, increasing the amount of noise to 30%, the total test error increases and becomes more balanced.

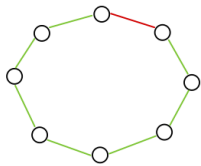

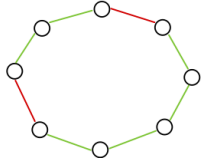

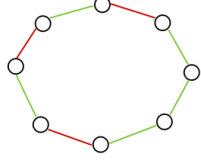

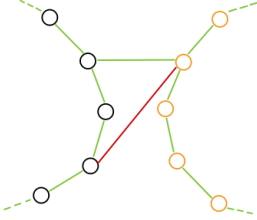
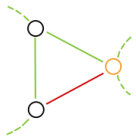
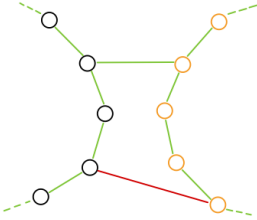
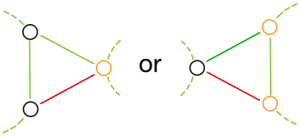
CONFIGURATION	COLLAPSED CONFIGURATION	TRANSITIVITY BREAKING
		YES
		No
		No
		YES
		YES

Figure 6. Left: Examples of input configurations among similar (first 3 lines) and different pairs (last two lines) in the presence of PLN. Center: configuration after similar pair collapse. If this can not be consistently reached, the minimal extended configuration (triangle with 1 red and 2 green edges) is displayed. Right: Presence of transitivity breaking coming from PLN.



## D. Additional Figures

### D.1. Double descent plots

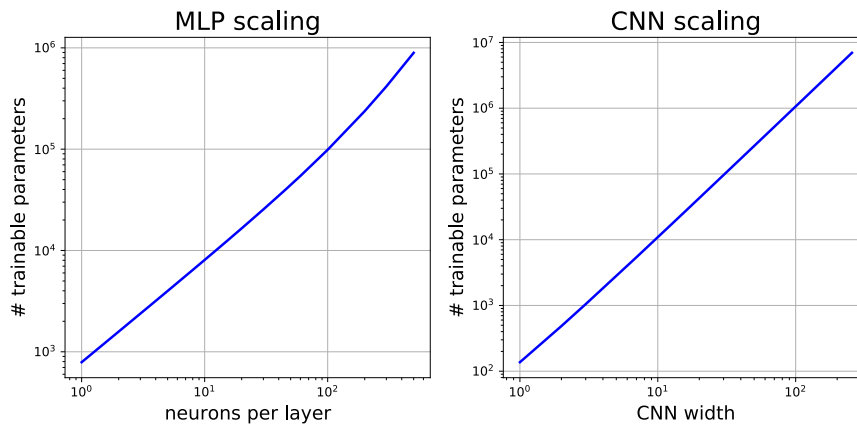


Figure 7. Number of trainable network parameters at increasing network size. The input shape is assumed to be (28,28,1). The structure of the MLP and the CNN is described in section 2.4.

MNIST - MLP - EUCLIDEAN DISTANCE

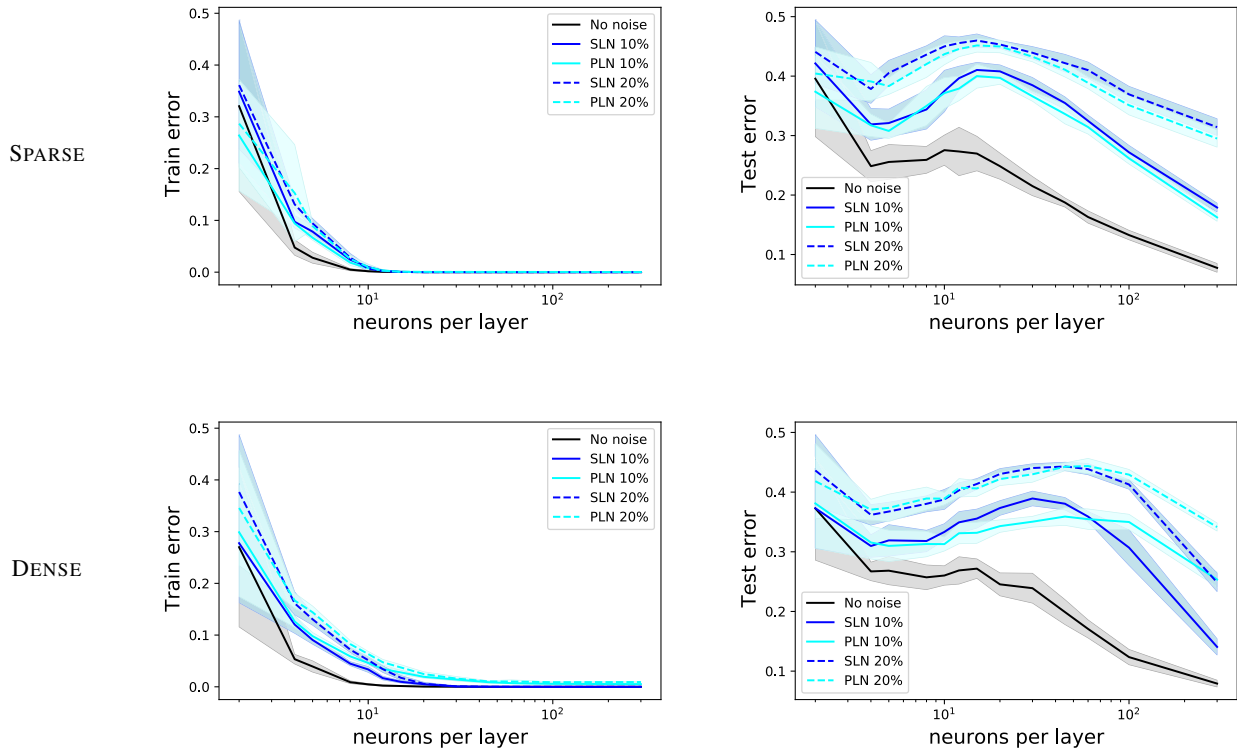


Figure 8. MNIST training (left) and test error (right) for a MLP with 3 layers of equal size after 2k epochs. We vary the number of neurons per layer and the effective PLN and SLN noise in the sparse and the dense configuration. We consider contrastive loss using the Euclidean distance between branch outputs, and we use 6k (9k) train (test) pairs.

FMNIST - MLP - EUCLIDEAN DISTANCE

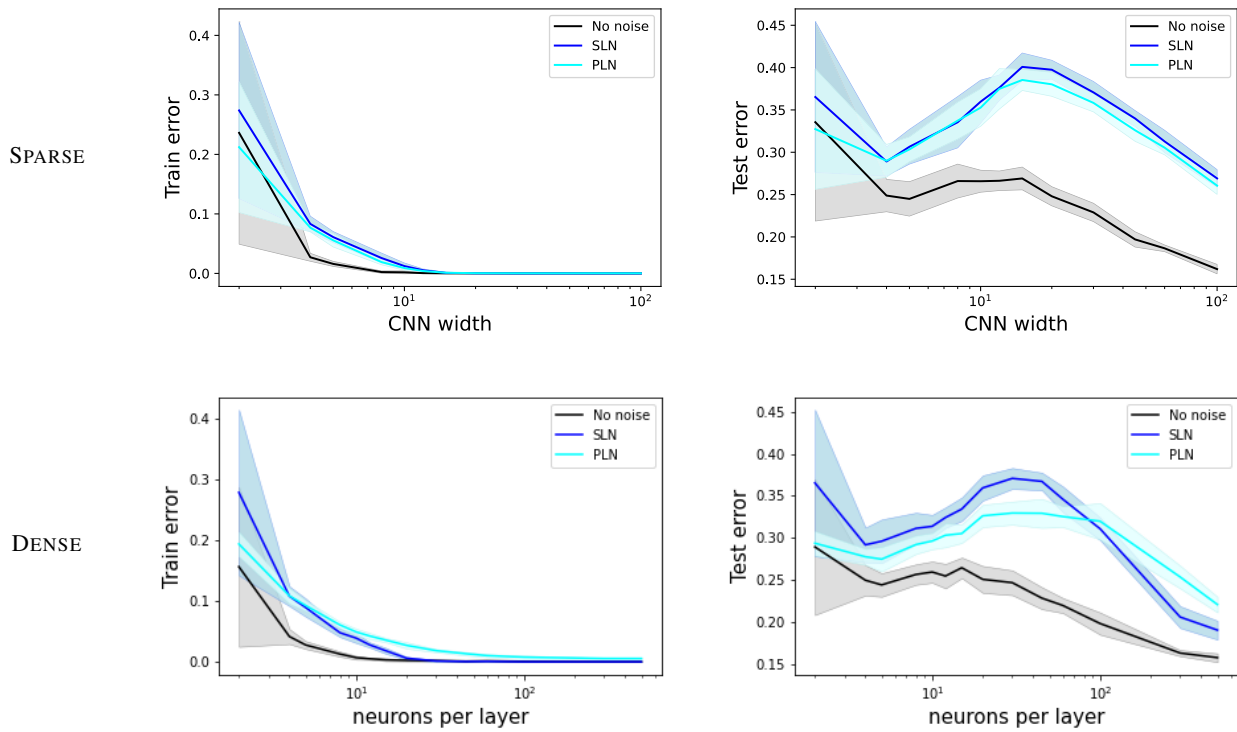


Figure 9. FMNIST training (left) and test error (right) for a for a MLP with 3 layers of equal size after 2k epochs. We vary the number of neurons per layer in the sparse and the dense configuration. We consider contrastive loss using the Euclidean distance between branch outputs, and we use 6k (9k) train (test) pairs.

MNIST - CNN - EUCLIDEAN DISTANCE

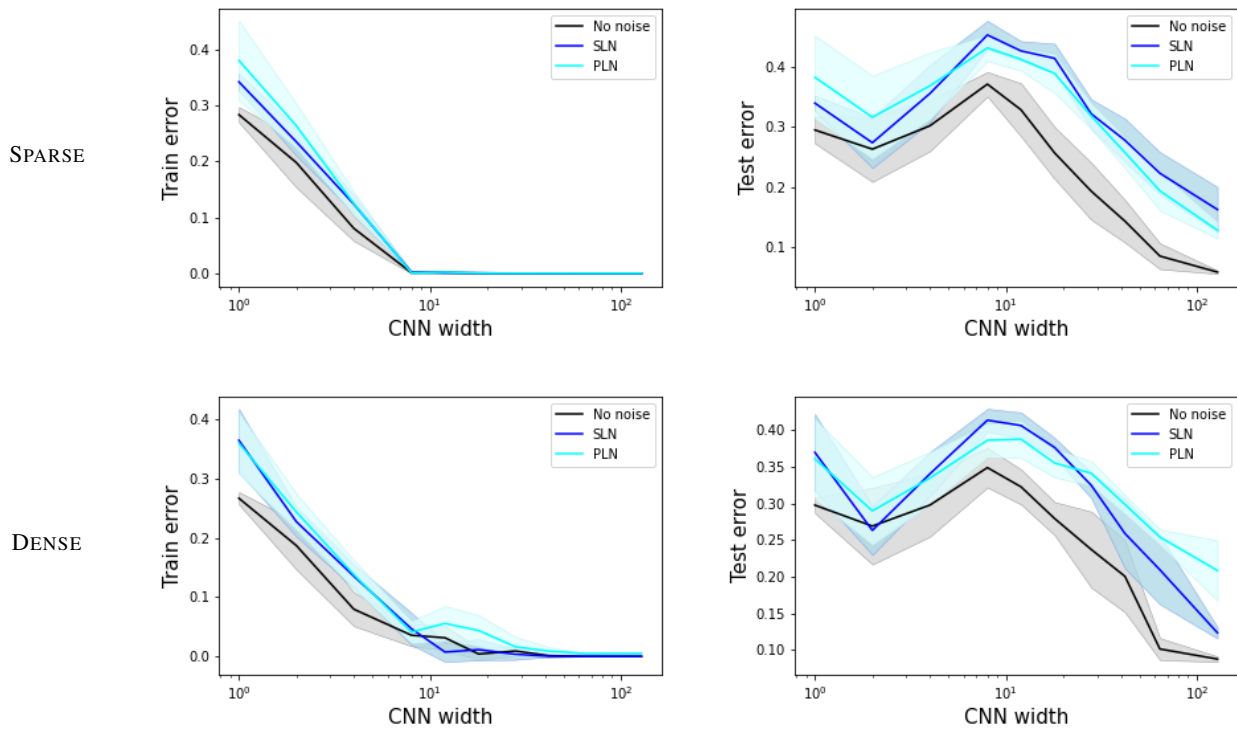


Figure 10. MNIST training (left) and test error (right) for a CNN with 4 layers of different size (see Section 2.4). We vary the CNN with parameter  $k$  in the sparse and the dense configuration of input pairs. We consider contrastive loss using Euclidean distance between branch outputs, and we use 6k (9k) train (test) pairs.

FMNIST - CNN - EUCLIDEAN DISTANCE

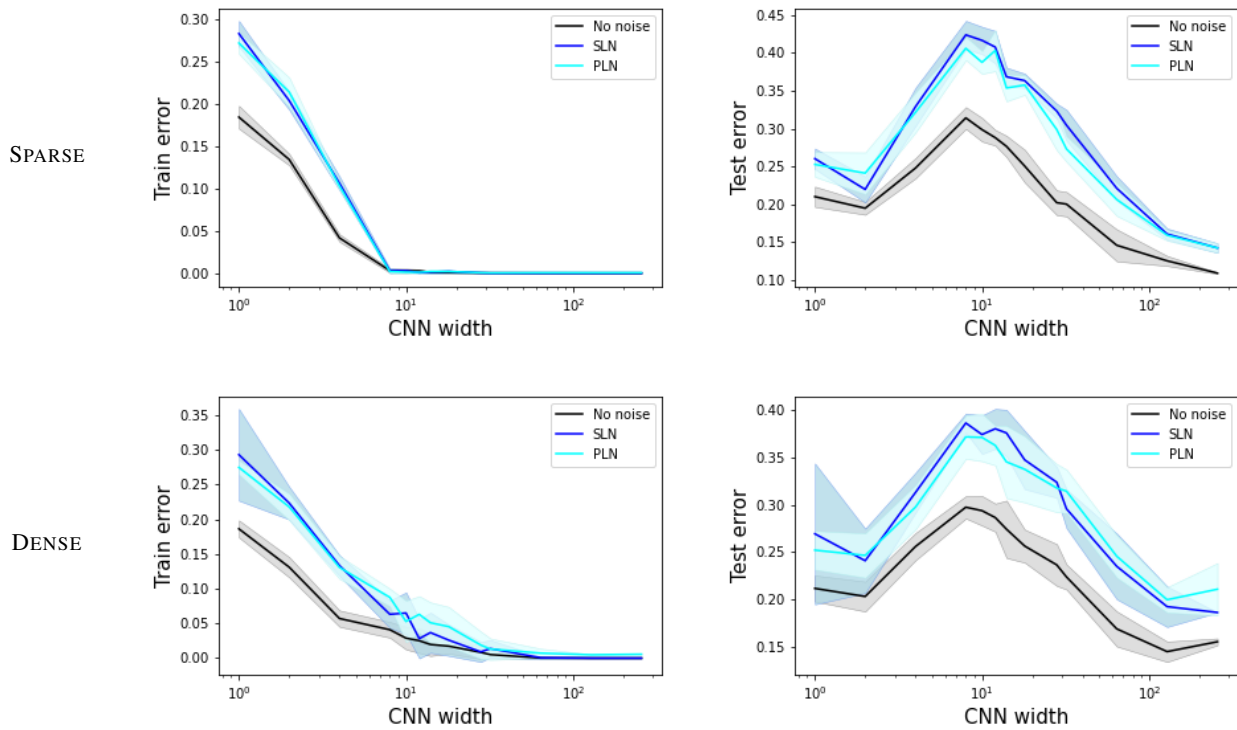


Figure 11. FMNIST training (left) and test error (right) for a CNN with 4 layers of different size (see Section 2.4). We vary the CNN with parameter  $k$  in the sparse and the dense configuration of input pairs. We consider contrastive loss using Euclidean distance between branch outputs, and we use 6k (9k) train (test) pairs.

MNIST - CNN - COSINE SIMILARITY

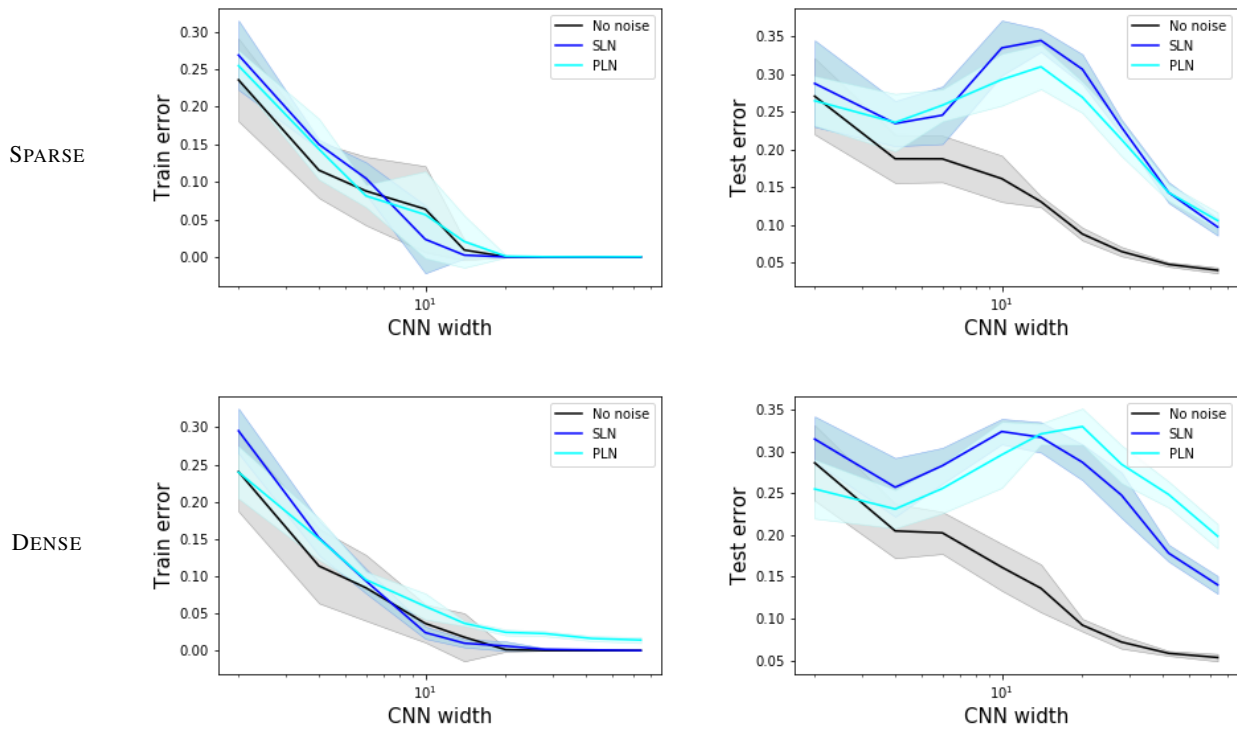


Figure 12. MNIST train (left) and test error (right) for a CNN with 4 layers of different size (see Section 2.4). We vary the CNN width parameter  $k$  in input pairs' sparse and dense configuration. We consider cosine embedding loss using cosine similarity between branch outputs, and we use 6k (9k) train (test) pairs.

FMNIST - CNN - COSINE SIMILARITY

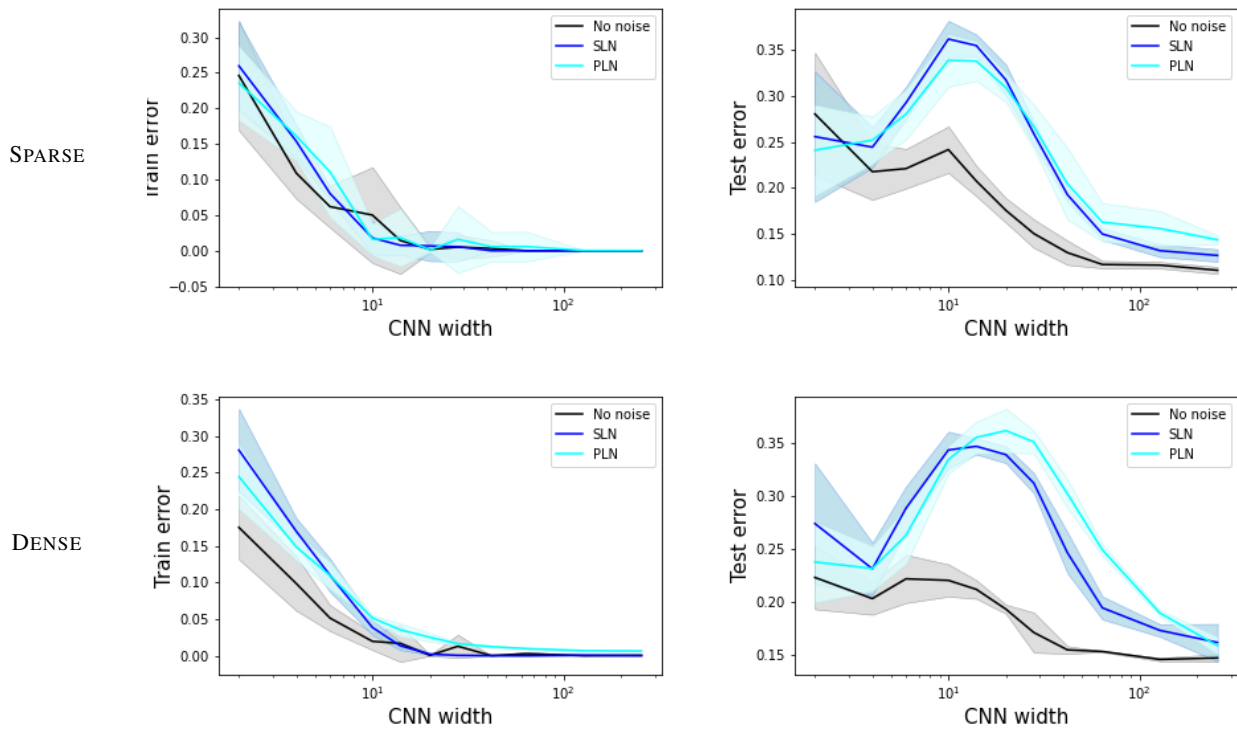


Figure 13. FMNIST train (left) and test error (right) for a CNN with 4 layers of different size (see Section 2.4). We vary the CNN width parameter  $k$  in input pairs' sparse and dense configuration. We consider cosine embedding loss using cosine similarity between branch outputs, and we use 6k (9k) train (test) pairs.

### D.2. Comparison between online and offline settings

Here we present several comparisons between online and offline settings for different metrics.

In Fig. 14, we compare the Ideal World without noise and in the presence of different percentages of single label noise (SLN) and pair label noise (PLN). As expected, test error gets worse as label noise increases, but the model still improves generalization with noisy data. In Fig. 15, we present the losses corresponding to the cases in Fig. 4. Figure 16 shows the Real and Ideal Losses for the sparse scenario (analogous to Figs. 4 and 15 for the dense case). Note that the Real Test Losses follow the same behavior as their corresponding Test Errors. We show in Fig. 17 the Real/Ideal comparison in the absence of label noise (the same as left-panel of Fig. 3), where the final training errors are indicated by the stars (dots) for the dense (sparse) scenario.

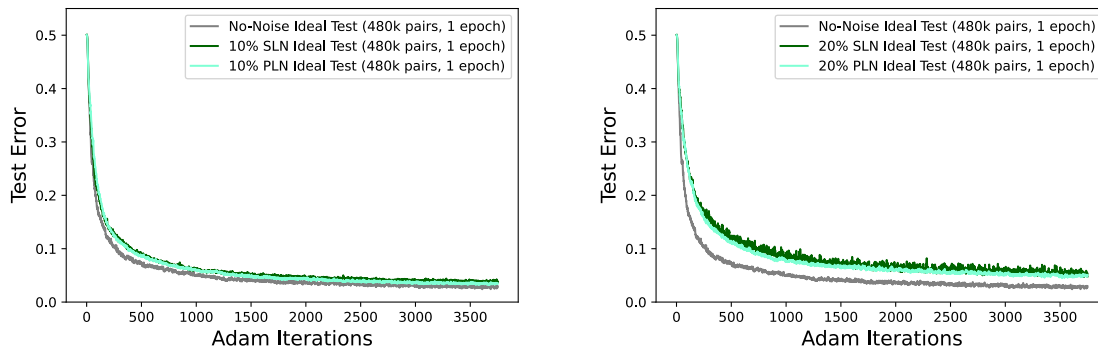


Figure 14. Comparison between Ideal Worlds for different levels of noise. Plots show the Test Errors as a function of the minibatch Adam iterations. We plot the median over 5 trials for the scenarios with SLN and PLN label noise (10% (left) and 20% (right)) for the MLP architecture with 200 nodes per layer. The experimental setup is described in Sec. 2.4.

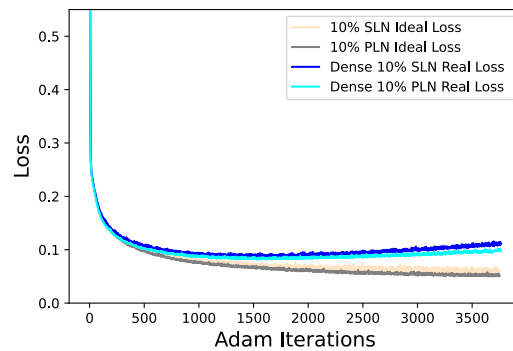


Figure 15. Ideal vs. Dense Real worlds with 10% of label noise for the MLP architecture with 200 nodes per layer. The plot shows Test Losses as a function of minibatch Adam iterations. The corresponding test errors are given in Fig. 4.



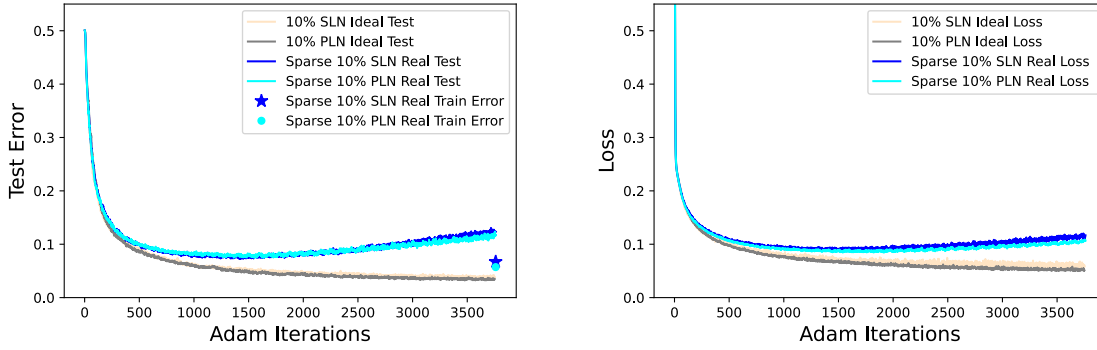


Figure 16. Ideal vs. Sparse Real worlds with 10% of label noise for the MLP architecture with 200 nodes per layer. Plots show the Test Errors (left) and Test Losses (right) as a function of minibatch Adam iterations. The star (dot) on the left corresponds to the SLN (PLN) Real World Train Error at the end of training.

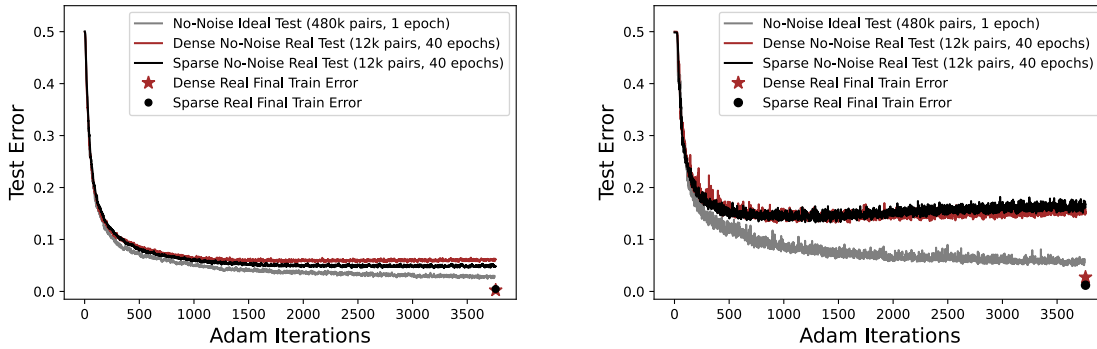


Figure 17. Ideal vs. Dense/Sparse Real worlds in the absence of label noise. Plots show Test Errors as a function of minibatch Adam iterations for the MLP (left) and CNN cases (right), as in the left-panel of Fig. 3. The stars (dots) correspond to the Dense (Sparse) Real World train error at the end of training.