THE FOES OF NEURAL NETWORK'S DATA EFFICIENCY AMONG UNNECESSARY INPUT DIMENSIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Input dimensions are unnecessary for a given task when the target function can be expressed without such dimensions. Object's background in image recognition or redundant sentences in text classification are examples of unnecessary dimensions that are often present in datasets. Deep neural networks achieve remarkable generalization performance despite the presence of unnecessary dimensions but it is unclear whether these dimensions negatively affect neural networks or how. In this paper, we investigate the impact of unnecessary input dimensions on one of the central issues of machine learning: the number of training examples needed to achieve high generalization performance, which we refer to as the network's data efficiency. In a series of analyses with multi-layer perceptrons and deep convolutional neural networks, we show that the network's data efficiency depends on whether the unnecessary dimensions are *task-unrelated* or *task-related* (unnecessary due to redundancy). Namely, we demonstrate that increasing the number of *task-unrelated* dimensions leads to an incorrect inductive bias and as a result degrade the data efficiency, while increasing the number of *task-related* dimensions helps to alleviate the negative impact of the *task-unrelated* dimensions. These results highlight the need for mechanisms that remove *task-unrelated* dimensions, such as crops or foveation for image classification, to enable data efficiency gains.

1 INTRODUCTION

The success of Deep Neural Networks (DNNs) contrasts with the still distant goal of learning in a data efficient manner, as for biological systems (Lake et al., 2015; Botvinick et al., 2019). The capability to learn with fewer training examples and achieve high generalization performance is key to solving more complex tasks and reducing application's costs. One of the main obstacles to develop data-efficient DNNs is the difficulty of understanding the aspects that cause the need for large amount of training examples.

Little is known about the effects of the input data representation to the DNN's data efficiency. A striking property of most datasets is that the data representation contains dimensions that are unnecessary to perform the task, *ie.,* the target function can be expressed without such unnecessary dimensions. The conventional wisdom is that the network learns to capture the useful patterns in the dataset and discards the unnecessary input dimensions. Examples of unnecessary input dimensions are image regions or sentences in a text that are unrelated and/or redundant to the task at hand. This begs the question: is more data needed to learn to discard such unnecessary input dimensions and what neural mechanisms does the network use to do so? In this paper, we answer these questions by analysing two distinct types of unnecessary input dimensions: *task-unrelated* and *task-related* dimensions. The former are always unnecessary and the latter are unnecessary only when they are redundant.

Our analysis builds on theoretical results that we derive for linear networks with square loss with unnecessary dimensions based on Gaussian noise (*task-unrelated*) and linear combinations of other dimensions (*task-related*). These results are extended by providing empirical evidence on synthetic and natural datasets, with more general types of unnecessary input dimensions such as context in object recognition and redundant sentences in text classification. We use Multi-Layer Perceptrons (MLPs) and Deep Convolutional Neural Networks (DCNNs). We conclude that increasing the number of *task-unrelated* dimensions degrades the data efficiency, while increasing the number of *task-related* dimensions helps to alleviate the negative impact of the *task-unrelated* dimensions. Also, we observe that DCNNs are affected by unnecessary dimensions in similar ways as the aforementioned cases. Furthermore, an analysis of the neural activity reveals that the neurons in all layers learn to discard the unnecessary features by responding only to the target object.

These results add to a growing body of literature that analyses the effect of object's background in image recognition and more generally our results add to the ongoing efforts to understand the generalization abilities of DNNs with respect to the dataset. In the following, we position our work in these two strands of research:

Impact of Object's Background Previous works have shown that DNNs for image recognition fail to classify objects in novel and uncommon backgrounds (Choi et al., 2012; Sun & Jacobs, 2017; Dvornik et al., 2018; Beery et al., 2018; Volokitin et al., 2017). Remarkably, popular object recognition datasets are biased to such an extent that DNNs can predict the object category even when the objects is removed from the image (Zhu et al., 2017). Barbu et al. (2019) introduced a new benchmark which addresses the biased co-occurrence of objects and background, among others, such as bias in the object's viewpoint. DNNs exhibit large performance drops in this benchmark compared to ImageNet (Deng et al., 2009). Recently, Borji (2020) has shown that a large portion of the performance drop comes from the bias in object's background, as classifying the object in isolation substantially alleviates the performance drop. Also, there is a plethora of methods based on data augmentation and transfer learning that can help alleviating the bias with object's background (Shorten & Khoshgoftaar, 2019).

In contrast to previous works, we analyse the impact of the object's background to the DNN's data efficiency when the dataset is unbiased, *ie.,* the statistics of the object's background are similar between training and testing and there is no bias in the co-occurrence of object and background. To the best of our knowledge, our work is the first to report that the presence of object's background harms the data efficiency of the network when the object's background is unbiased.

Overparametrization and Data Dimensionality A remarkable characteristic of DNNs is that their accuracy does not degrade when the network's width is increased by adding more hidden units, despite overparameterization (*ie.,* the network has a larger number of parameters than training examples that can lead to overfitting) (Poggio et al., 2017; Novak et al., 2018). Adding unnecessary input dimensions is another way to increase the width and the number of parameters of the network. Yet, we show that the effect of overparameterization by increasing the amount of hidden units is different from overparameterization by increasing the unnecessary input dimensions, as the latter degrades the accuracy when the dimensions are *task-unrelated*. This finding establishes a boundary of the DNNs' robustness to overparametrization.

Another strand of research relates the structure of the dataset with the generalization ability of the network. Several works in statistical learning theory for kernel machines relate the spectrum of the dataset with the generalization performance (Zhang, 2005). For neural networks, Recanatesi et al. (2019); Ansuini et al. (2019) define the intrinsic dimensionality based on the dimension of the data manifold. These works analyse how the network reduces the intrinsic dimension across layers. Yet, these metrics based on manifolds do not provide insights about how specific aspects of the dataset, *e.g.,* unnecessary dimensions, contribute to the intrinsic dimensionality.

2 Unnecessary Input dimensions on linear networks and MLPs

In this section we introduce theoretical results for linear network that serve as a basis for the rest of the analysis. Then, we report results with linear networks and MLPs without the assumptions made for the theoretical development.

2.1 Exact solution for linear networks

We use (X, Y) to denote the dataset, which we assume to be generated through a linear function, where $X = [\mathbf{x}_1 \cdots \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ contains n independent observations of p uncorrelated features, and $Y = [\mathbf{y}_1 \cdots \mathbf{y}_n] \in \mathbb{R}^{o \times n}$ denotes the respective output. We define p as the minimal dimensionality. We aim at estimating the function f^*, such that, for every example $(\mathbf{x}_i, \mathbf{y}_i)$, $\mathbf{y}_i = f^*(\mathbf{x}_i)$ holds approximately. The choice of a linear network $f(\mathbf{x}) = W\mathbf{x}$, with $W \in \mathbb{R}^{o \times p}$ and the square loss as cost function assures that, even for $p > n$ (overparameterized case), optimization through gradient descent leads to a unique solution that corresponds, among all, to the one with minimum norm, namely the pseudo-inverse (Schölkopf et al., 2001):

$$W^{+} = Y(X^{\top}X)^{-1}X^{\top}, \text{ with } p > n. \tag{1}$$

Given a dataset, we increase the amount of input dimensions by adding *task-related* dimensions, *task-unrelated* dimensions, or a combination of both, *task-related/unrelated* dimensions. We concatenate those to the p original dimensions and predict the effect on the generalization performance. In the following, we introduce the theoretical results for each case separately assuming the network operates in the overparametrized regime, $p > n$, which is the most common in practice:

Task-related dimensions Let $T \in \mathbb{R}^{d \times p}$ be the linear transformation which we apply on the vector of p minimal dimensions to generate unnecessary redundant dimensions. The dataset with unnecessary dimensions is the result of the transformation $F = [\mathbb{I}_p^{\top} \; T^{\top}]^{\top} \in \mathbb{R}^{(p+d) \times p}$, where $\mathbb{I}_p$ denotes the identity matrix of size p. The solution for the overparameterized case corresponds to

$$W^{+} = Y(X^{\top}F^{\top}FX)^{-1}X^{\top}F^{\top}. \tag{2}$$

We assume F is a tight frame (Daubechies, 1992), *ie.,* there is a unique scaling factor $a > 0$ such that $\|F\mathbf{v}\|^2 = a\|\mathbf{v}\|^2$ holds for any vector $\mathbf{v} \in \mathbb{R}^p$. Thus, Eq. 2 corresponds to $W^{+} = a^{-1}Y(X^{\top}X)^{-1}X^{\top}F^{\top}$, which is the same as the linear network learnt without *task-related* dimensions in Eq. 1 (the scaling constant a compensates with the term at the numerator at prediction).

Task-unrelated dimensions For each example i, we denote $\mathbf{n}_i \in \mathbb{R}^d$ a vector of d *task-unrelated* dimension independent from $\mathbf{x}_i \in \mathbb{R}^p$. The new input vector is $[\mathbf{x}_i^{\top} \; \mathbf{n}_i^{\top}]^{\top}$. Intuitively, if $\mathbf{n}_i^{\top}$ is randomly generated, it is expected that as the *task-unrelated* dimensions increase there is more overfitting. Yet, there are exceptions to this intuition, *e.g.,* when the labels of the dataset are noisy. To illustrate this point, we consider *task-unrelated* dimensions distributed as a Gaussian, with zero mean, diagonal covariance and same variance σ^2: $(\mathbf{n}_i)_\ell \sim \mathcal{N}(0, \sigma^2)$, $\ell = 1, \ldots, d, i = 1, \ldots, n$. In this case, the following approximation of the prediction of the linear network holds for very large d:

$$W^{+}[\mathbf{x}_{\text{test}}^{\top} \; \mathbf{n}_{\text{test}}^{\top}]^{\top} \simeq Y(X^{\top}X + d\sigma^2\mathbb{I}_n)^{-1}X^{\top}\mathbf{x}_{\text{test}}, \tag{3}$$

where $[\mathbf{x}_{\text{test}}^{\top} \; \mathbf{n}_{\text{test}}^{\top}]^{\top}$ is an arbitrary test sample (see App. A.1 for the details). We can observe by analysing Eq. 3 that $d\sigma^2\mathbb{I}_n$ corresponds to the Tikhonov regularization term, with regularization parameter $\lambda = d\sigma^2$. Thus, the presence of a large number of *task-unrelated* Gaussian dimensions is equivalent to a network trained on the dataset without *task-unrelated* dimensions, biased towards small ℓ_2 norm weights.

Combining task-related and task-unrelated dimensions Let d be the total number of unnecessary input dimensions. Among those, a percentage $\nu \in [0, 1]$ are *task-related* and the rest are *task-unrelated.* The aforementioned assumptions for the unnecessary dimensions lead to a solution affected by the ℓ_2 bias, with respect to the minimal dimensions problem. This effect can mitigated by the presence of *task-related* dimensions. See App. A.2 for further details.

2.2 Results of linear networks

We now evaluate the linear networks on a regression and a classification problem, following the assumptions above. For the regression experiment, we generate a dataset using a linear teacher network $\mathbf{y}_i = W^{*}\mathbf{x}_i$, with $W^{*} \in \mathbb{R}^{2 \times 30}$ (*ie.,* $o = 2, p = 30$) components, generated from a standard distribution. The *task-unrelated* dimensions have $\sigma = 0.1$, while *task-related* dimensions are linear combinations of the p (= 30) minimal dimensions of the dataset through a tight frame. For the classification experiment, we take a regression network, as for the regression case, and we quantize the output to two categories. In both regression and classification, we analyse the effects of a number of unnecessary features between 2 to 500. We repeat the classification experiments 10 times. In the cases in which the problem becomes underparameterized, we resort to its correspondent closed form solution, otherwise we use the pseudoinverse in Eq. 1.

In Fig. 1 and Fig. 2 we show the predictive performance as we increase the amount of unnecessary dimensions. In the following, we introduce the main take-aways from this experiment:

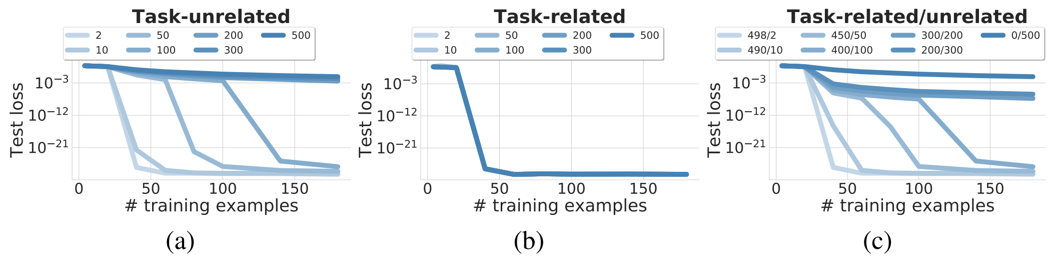


Figure 1: Test loss for the regression task with a linear network: (a) task-unrelated, (b) task-related, and (c) task-related + task-unrelated cases. The legends report the amount of unnecessary dimensions.

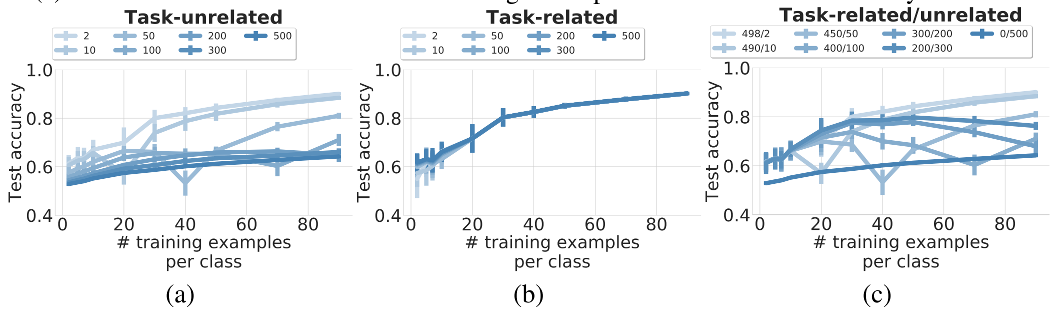


Figure 2: Test accuracy for classification task with a linear network: (a) task-unrelated, (b) task-related, and (c) task-related + unrelated cases. Plots (a) and (c) show the double descent phenomenon.

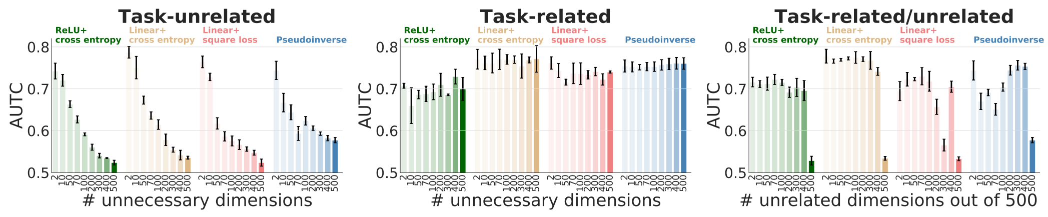
Increasing the amount of *task-unrelated* dimensions leads to a regularization effect that could be beneficial or not The results for the *task-unrelated* dimensions are shown in Fig. 1a and Fig. 2a. We can observe that the general trend is that there is more overfitting (*ie.,* worse data efficiency) as the number of *task-unrelated* dimensions is increased. Recall that for a large amount of *task-unrelated* dimensions the linear network is equivalent to the one trained on the minimal dimensions of the dataset with Tikhonov regularization (Eq. 3). Thus, the results show that the regularization is having a negative effect (*ie.,* the regularization leads to underfitting with respect to the minimal dimensions of the dataset, which is equivalent to overfitting to the dataset with the unnecessary dimensions).

In Fig. 2a we also observe that depending on the number of training examples, a small number of *task-unrelated* dimensions may lead to more overfitting than for the large number of *task-unrelated* dimensions. The peaks of overfitting observed are due to the double descent phenomenon (Advani & Saxe, 2017; Belkin et al., 2019; Nakkiran et al., 2020; Poggio et al., 2019). Classification of linearly separable datasets using the square loss function exhibits poor generalization when the number of training examples is near to the number of independent components (Advani & Saxe, 2017). This phenomenon is more prominent when there is noise in the dataset labels, which is the case for our classification dataset as it comes from quantizing the regression dataset. Note that for large amounts of *task-unrelated* dimensions, the double descent phenomenon is not present, as the regularizer helps avoiding the peak of overfitting. We further investigate the advantages of the induced Tikhonov regularization in regression by adding noise in target output. In App. B.1, through a dataset generated as a proof of concept, we show that, for a large number of *task-unrelated* dimensions, the prediction error coincides with the Tikhonov regularization term, and leads to an improvement.

Increasing the amount of *task-related* dimensions does not affect the linear network and helps alleviating the effect of the *task-unrelated* dimensions In Fig. 1b and Fig. 2b, we corroborate that *task-related* dimensions lead to the same linear network. Furthermore, in Fig. 1c and Fig. 2c, we observe that the increase of *task-related* dimensions improves the signal to noise ratio, with a further positive impact on performance: the *task-related/unrelated* case for $300/200$ in Fig. 2c shows much higher performance and data efficiency than the correspondent experiment with $d = 300$ in Fig. 2a.

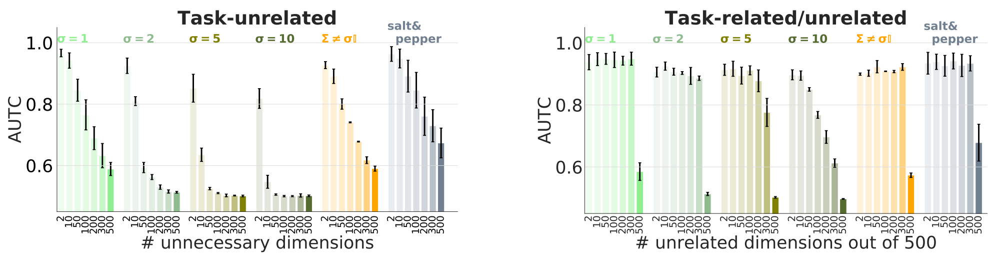
2.3 Results of MLPs

We now show that the take-aways obtained with linear networks extend to MLPs. For each of the aforementioned theoretical assumptions, we run an experiment to evaluate the impact it has on the



(a) (b) (c)

Figure 3: AUTC curves for different MLPs on the classification task: (a) task-unrelated, (b) task-related, and (c) task-related + unrelated case. In green MLP with ReLU non-linearity and cross entropy loss, in beige and pink linear MLPs with cross entropy and mean square error losses respectively, in blue exact solution from Figure 2.



(a) (b)

Figure 4: AUTC curves for the MLP with non-linear ReLU activations and cross entropy loss on the mixed Gaussians dataset, for different distributions of task-unrelated dimensions.

conclusions when we drop the assumption. In order to facilitate displaying the results across a series of experiments, we use from now on the Area Under the Test Curve (AUTC) as a measure of the generalization performance for different training set size (see App. B.4 for details).

Non-linear networks In the classification dataset introduced above, we evaluate MLPs with one hidden layer consisting of 128 nodes. We evaluate different variations of the MLP depending on the non-linearities and the loss function: ReLU with soft-max cross-entropy, linear with soft-max cross-entropy or square loss. We report in Fig. 3 the results across 3 repetitions. “Pseudoinverse” refers to the solution computed using the theoretical solution from the previous section, whose results were reported in Fig. 2 and in Fig. 3 correspond to blue bar plot. The AUTC for the different MLPs shows that the data efficiency follows similar trends as the theoretical solutions as we increase the amount of unnecessary input dimensions, although the different variants differ in terms of overall data efficiency. Namely, as the amount of *task-unrelated* dimensions is increased the data efficiency decreases (Fig. 3a), while *task-related* dimensions do not negatively affect the data efficiency (Fig. 3b) and can even compensate the the negative effects of the *task-unrelated* dimensions (Fig. 3c). Finally, note that for some amount of *task-unrelated* dimensions in Fig. 3c, the AUTC of linear MLP with square loss has a similar behavior of the pseudoinverse solution, for different task-unrelated dimensions. We suspect that this behavior is a consequence of optimization.

Non-linearly-separable datasets To further test the generality of previous results on data distributions that are not linearly separable, we use a mixture of Gaussians to generate non-linearly-separable datasets for binary classification. Each class consists of three multivariate Gaussians of dimensions $p = 30$. As we fix a class, we generate a sample by randomly selecting, among the three, one distribution from which we draw the minimal dimensions. We evaluate the MLP with ReLU and soft-max with cross-entropy loss because among the different variants it is the only well suited to fit non-linearly-separable data. We report the results in App. B.3, and these are consistent with the previous conclusions. We observe that as the dataset becomes more non-linearly-separable, the data efficiency drops, which is expected.

Table 1: Filters and max-Pooling sizes at the first layer for DCNNs trained on Natural MNIST.

	Adapt	FP 256	FP 128	FP 64	FP 28
filter & pooling sizes	$r \cdot 3$, $r \cdot 2$	27, 18	14, 9	7, 5	3, 2

Beyond Gaussian distributed *task-unrelated* input dimensions We evaluate the effect of the variance of the Gaussian distribution to generate the unnecessary dimensions and we also evaluate two other types of synthetic noise, namely, Gaussian noise with $\Sigma_{ii} = 1,\ \forall i$, with $\Sigma_{ij} = 0.5,\ \forall i \neq j$, and salt and pepper noise, where each vector component can assume value (0, or u), based on a Bernoullian distribution on $\{-1, 1\}$. We show the results for the ReLU MLP with soft-max cross-entropy loss in Fig. 4. We observe the same trends as all previous experiments. For *task-unrelated* dimensions, data efficiency deteriorates as the variance of Gaussian noise increases.

3 Unnecessary input dimensions on DCNNs

In this section we analyze the data efficiency of DCNNs in image and text classification tasks. The weight sharing of the convolutional layers in DCNNs facilitate learning in problems where both the position of the dimensions that are necessary for the target function and the amount of the unnecessary dimensions are not fixed, *e.g.,* object recognition with objects at different image positions. In the following we first analyze the data efficiency for the unnecessary input dimensions fixed as we have done for linear networks and MLP, and then, for non-fixed position and number of unnecessary input dimensions in realistic datasets.

3.1 Fixed unnecessary input dimensions by centered object

We generate two datasets based on the MNIST dataset (LeCun et al., 1998): Synthetic MNIST and Natural MNIST. In both cases, the digit is always positioned at the center of the image, and we use different types of backgrounds for adding *task-unrelated* dimensions. We build Synthetic MNIST in analogy to the MLP experiments. In the *task-unrelated* case, we add random pixels at the edge of the image, while for the *task-related* version, we upscale the MNIST digit to a different sizes. In the *task-related/unrelated* case, we fix the size of the image and we change the ratio of *task-related* and *unrelated* dimensions by upscaling the MNIST digit. The Natural MNIST consists on the MNIST digit at the center of a natural scene from the Places dataset (Zhou et al., 2014), following the procedure by Volokitin et al. (2017). We reduce the amount of unnecessary dimensions by upscaling the MNIST edge to four different sizes. See App. C.1 for details about the data generation.

We use a DCNN model with three convolutional layers followed by two fully connected layers. Details about networks hyper-parameters are in App. C.2. We test our models across different configurations of Filter (F) and max-pooling (P) sizes at the first layer, as reported in Tab. 1. We also evaluate adapting these parameters to the ratio between the object edge and the original MNIST edge (28), which we refer as r.

DCNNs suffer from unnecessary input dimensions but less than MLP First, we compare the data efficiency of MLPs and DCNNs on the Synthetic MNIST using the same training protocol (App. C.3). DCNNs are trained and tested in the "adapted" configuration shown in Tab. 1. In Fig. 5, we report the log AUTC (App. B.4) and as expected, we observe that DCNNs have higher generalization performance than MLPs across the three scenarios. Yet, the data efficiency of DCNN suffers from *task-unrelated* input dimensions, while it does not get affected by an increase of *task-related* dimensions.

We compare different DCNN architectures on the Natural MNIST dataset (details on the training setup are in App. C.4). Fig. 6a, on the left, shows the mean log AUTC across different models (top) and the accuracy for 20 training examples (bottom), across three repetitions of the experiment. Each group of bar plots refers a different filter and max-pooling choice at the first layer, as shown in Tab. 1. As for Synthetic MNIST and all previous experiments, we observe that for most architectures the accuracy for small amount of examples decreases as the number of *task-unrelated* dimensions increases, and a large gap among architectures emerges as the number of training examples per class (n_{tr}) decreases. This gap increases when the networks have a global average-pooling before the

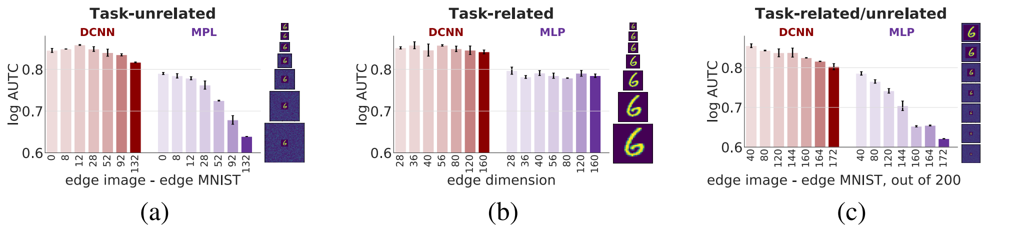


Figure 5: log AUTC curves of DCNNs and MLP with ReLU and cross entropy loss on Synthetic MNIST datasets. The log AUTC is computed in the range $n_{tr} \in [1, 10^3]$. For cases (a)-(c), on the right, examples for each dataset.

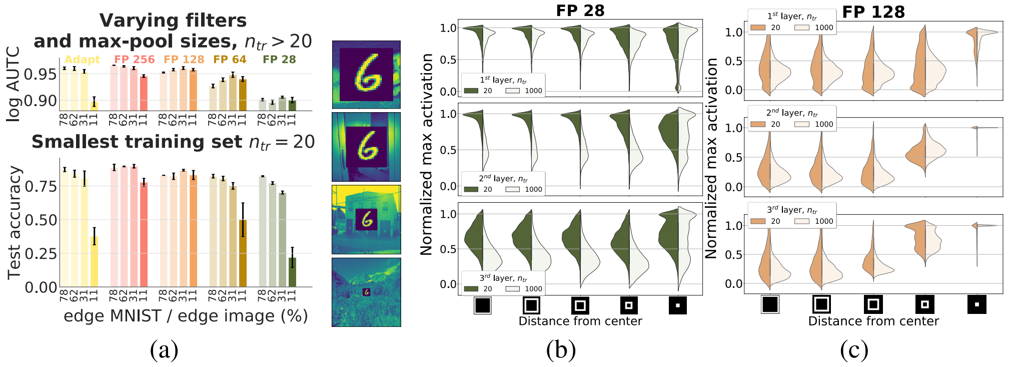


Figure 6: Natural MNIST. (a) top: log AUTC of DCNNs across different filters and max-pooling sizes at the first layer computed on n_{tr} in the range [50, 1000]. (a) bottom: test accuracy across DCNNs for a number of training examples $n_{tr} = 20$ and different amount of unnecessary input dimensions. (a) right: examples for different amount of unnecessary input dimensions, from the top to the bottom, the edge MNIST sizes are 200, 150, 80, and 28. (b-c) distribution of max activation values, as we move from the edges of the hidden representations of test images to their centers, across the three convolutional layers.

fully connected layers (see Fig. 12 in App. C.6), despite the decrease of the difference in amount of free parameters among models for the global pooling case. The optimal architectural choices are dataset dependent and these results highlight that such choices have an important impact to the data efficiency.

Activation analysis reveals high tuning to discard object's background We now analyse the emergent mechanisms in the DCNN that facilitate discarding the unnecessary dimensions. We verify a simple hypothesis that the kernels become tuned to discard the object's background and lead to activity only for the object. We provide evidence that supports this hypothesis for networks trained on the smallest $n_{tr} = 20$ and largest $n_{tr} = 1000$ training sets with the largest amount of unnecessary *task-unrelated* dimensions, such that we can better capture the differences in the neural activity as the network is trained with more examples. We focus on the networks FP 28 and FP 128, respectively which are the most data efficient and data inefficient architectures we tried. For these models, at each layer, we extract the neural activity of 1000 test samples after the max-pooling. We grouped the representation values at each layer depending on their distance from the center. Figs. 6b and c show the distribution of normalized maximum activity for each kernel as we go from the edges to the center of the neural activity feature maps. Figs. 6b and c show that when the network is trained with more examples (light violin plots), the neurons respond more to the object compared to the background. Yet, when the networks are trained with few examples (dark violin plots), the differences in terms of neural activity are minor between the object and its background. Note that this phenomenon is much more pronounced for the most data efficient architecture (FP 128). These results suggests that the data efficiency of the network is driven by the emergence of kernels that can detect the object while not responding to the background.

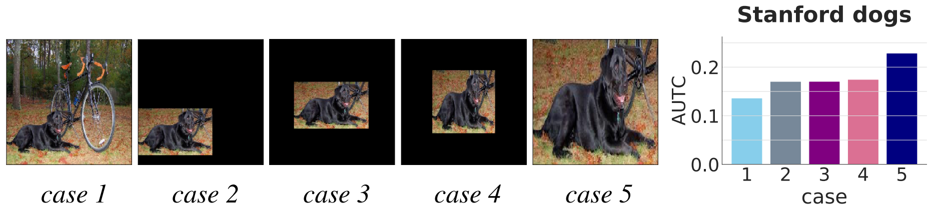


Figure 7: Accuracy on Stanford Dogs with different amount of unnecessary dimensions. An example for each of the five cases given as input to ResNet-18 (left). AUTC curves of ResNet-18 models trained on the fives cases (right).

3.2 Non-fixed dimensions

Objects appear at different scales and positions across images. We analyze the effect of unnecessary dimensions when these are in non-fixed positions and amount. We evaluate on the dog breed classification benchmark called Stanford Dogs dataset (Khosla et al., 2011). The dataset has 100 examples for each of 120 dogs breeds, for a total of 12000 training images and 8580 test images. Each image in the original dataset is annotated with information about the dog's position through bounding-box. We use the bounding-boxes provided in the dataset to determine the dimensions that are *task-unrelated*. Recall that one of the premises of this work is that there is no bias in the unnecessary dimensions. It is reasonable to expect that this is the case for this dataset, as dog breeds likely do not co-occur on similar backgrounds and hence, the network can not use background cues to determine the dog's category.

In Fig. 7, we compare the performance of a ResNet-18 (He et al., 2016) across five different versions of the training set. *Case 1* corresponds to the original image, with background and dogs at different scales and positions in the image. For *case 2* we remove the background, by multiplying the pixels outside of the bounding box to zero. In *case 3*, the dog in the bounding box is cropped and positioned at the center of the image, at its original scale. The other pixels have value zero. In *case 4* we rescale the dog to half of the final image edge, and we place this version of the dog at the center. Outer pixels have zero values. In *case 5* we crop the bounding box, and scale it to the final image size. We recenter the pixels values based on the standard procedure on ImageNet. (App. D). Fig. 7 left shows the AUTC values across the different cases, for different amounts of training examples per category. The use of the original dataset leads to the lowest data efficiency, while the highest data efficiency is achieved for *case 5*, when ResNet-18 is trained and tested on a cropped and rescaled image. The rest of the cases show that the DCNN needs more data to learn recognize the object at different position and scale, which is expected.

We run a final control to make sure that our conclusions apply to other domains. We use the text classification task of SST2 (Socher et al., 2013). The dataset suggests an heuristic to discard phrases that are unnecessary for the task. We compare DCNNs and MLPs for different amount of such unnecessary phrases, the results are shown in App. E, and corroborate the conclusions of the paper.

4 Conclusions

We have provided extensive evidence in synthetic datasets and object and text recognition benchmarks, for linear networks, MLPs and DCNNs to support the following conclusion: *task-unrelated* dimensions harm the data efficiency of neural networks while increasing the number of *task-related* dimensions may help alleviate the negative effect of *task-unrelated* dimensions. This conclusion builds on a theoretical analysis of a linear network trained with the square loss, which demonstrates that a large amount of Gaussian distributed *task-unrelated* dimensions are equivalent to Tikhonov regularization, while linear combinations of *task-related* dimensions serve to adjust for the amount of such regularization. Also, we have identified the mechanisms used by DCNNs to discard unnecessary dimensions, which consists on the emergence of neurons tuned to detect the target objects and to discard the unnecessary dimensions. These results highlight the need of introducing strategies to discard unnecessary dimensions to improve the DNN's data efficiency, suggesting that foveations or crops of the relevant image features, *e.g.,* (Luo et al., 2016), may lead to substantial gains of data efficiency for object recognition.

REFERENCES

Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 6109–6119, 2019.

Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in Neural Information Processing Systems*, pp. 9448–9458, 2019.

Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 456–473, 2018.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Ali Borji. ObjectNet dataset: Reanalysis and correction. *arXiv preprint arXiv:2004.02042*, 2020.

Matthew Botvinick, Sam Ritter, Jane X Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422, 2019.

Myung Jin Choi, Antonio Torralba, and Alan S Willsky. Context models and out-of-context objects. *Pattern Recognition Letters*, 33(7):853–862, 2012.

Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Nikita Dvornik, Julien Mairal, and Cordelia Schmid. Modeling visual context is key to augmenting object detection datasets. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 364–380, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. The dataset is available at `http://vision.stanford.edu/aditya86/ImageNetDogs/` (Last access: Oct. 1, 2020).

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, 2014.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. The dataset is available at `http://yann.lecun.com/exdb/mnist/` (Last access: Oct. 1, 2020).

Y. Luo, X. Boix, G. Roig, T. Poggio, and Q. Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv*, 2016.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR) Workshop*, 2013.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh Mhaskar. Theory of deep learning iii: explaining the non-overfitting puzzle. *Center for Brains, Minds and Machines (CBMM), Memo*, 2017.

Tomaso Poggio, Gil Kur, and Andrzej Banburski. Double descent in the condition number. Technical Report CBMM Memo No. 102, Center for Brains, Minds and Machines, 2019.

Stefano Recanatesi, Matthew Farrell, Madhu Advani, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Dimensionality compression and expansion in deep neural networks. *arXiv preprint arXiv:1906.00443*, 2019.

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *Proceedings of the International Conference on Computational Learning Theory (COLT)*, pp. 416–426, 2001.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1631–1642, 2013. The dataset is available at `https://nlp.stanford.edu/sentiment/` (Last access: Oct. 1, 2020).

J. Sun and D. W. Jacobs. Seeing what is not there: Learning context to determine where objects are missing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1234–1242, 2017.

Anna Volokitin, Gemma Roig, and Tomaso A Poggio. Do deep neural networks suffer from crowding? In *Advances in Neural Information Processing Systems*, pp. 5628–5638, 2017.

Tong Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17 (9):2077–2098, 2005.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using Places database. In *Advances in Neural Information Processing Systems*, pp. 487–495, 2014. The dataset is available at `http://places.csail.mit.edu/` (Last access: Oct. 1, 2020).

Zhuotun Zhu, Lingxi Xie, and Alan L Yuille. Object recognition with and without objects. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3609–3615, 2017.

A Exact form solution

A.1 Derivation of the Tikhonov solution Eq. 3

For the *task-unrelated* case, the problem is ill-posed as long as $p + d > n$ and the solution is

$$W^{+} = Y(X^{\top}X + N^{\top}N)^{-1}[X^{\top} \;\; N^{\top}]. \tag{4}$$

From this general solution, we compute the action of W^{+} on a generic test sample $[\mathbf{x}_{\text{test}}^{\top} \; \mathbf{n}_{\text{test}}^{\top}]^{\top}$ as

$$\begin{aligned} W^{+}[\mathbf{x}_{\text{test}}^{\top} \; \mathbf{n}_{\text{test}}^{\top}]^{\top} &= Y(X^{\top}X + N^{\top}N)^{-1}[X^{\top} \;\; N^{\top}][\mathbf{x}_{\text{test}}^{\top} \; \mathbf{n}_{\text{test}}^{\top}]^{\top} \\ &= Y(X^{\top}X + N^{\top}N)^{-1}(X^{\top}\mathbf{x}_{\text{test}} + N^{\top}\mathbf{n}_{\text{test}}). \end{aligned} \tag{5}$$

The second parenthesis expresses the correlation between each example from the training set with the new test example. More specifically, the parenthesis contains the sum of two contributions: the former measures the correlation by considering the matrix of the minimal dimensions only, and the latter quantifies the correlation originated by task-unrelated dimensions. We focus on the latter term, which we write in a more explicit form as

$$N^{\top}\mathbf{n}_{\text{test}} = \left[\mathbf{n}_1^{\top}\mathbf{n}_{\text{test}} \;\cdots\; \mathbf{n}_n^{\top}\mathbf{n}_{\text{test}}\right]^{\top} = \left[\sum_{\ell=1}^{d}(\mathbf{n}_1)_{\ell}(\mathbf{n}_{\text{test}})_{\ell} \;\cdots\; \sum_{\ell=1}^{d}(\mathbf{n}_n)_{\ell}(\mathbf{n}_{\text{test}})_{\ell}\right]^{\top}. \tag{6}$$

Remember our assumption that all $(\mathbf{n}_i)_{\ell}$ and $(\mathbf{n}_{\text{test}})_{\ell}$ are samples of two independent zero-mean Gaussian random variables with the same variance σ^2. For very large d values we can approximate the average realization of a random variable through its expected value, due to the law of large numbers. Here, by considering the sum, we multiply the expected value by d and we obtain

$$\begin{aligned} \sum_{\ell=1}^{d}(\mathbf{n}_i)_{\ell}(\mathbf{n}_{\text{test}})_{\ell} &\simeq d\mathbb{E}[\text{product of two independent zero-mean Gaussian random variables}] \\ &= d\left(\mathbb{E}[\text{zero-mean Gaussian random variable}]\right)^2 = 0. \end{aligned} \tag{7}$$

Given the independence of the two random variables we consider the product of their expectations, which is null for zero-mean distribution. Therefore, for any test example, the approximation

$$N^{\top}\mathbf{n}_{\text{test}} \simeq \mathbf{0} \tag{8}$$

holds.

Similarly, for the correlation term related to task-unrelated dimensions, in the first parenthesis of Eq. 5, we have, for $i \neq j$

$$\begin{aligned} \mathbf{n}_i^{\top}\mathbf{n}_j = \sum_{\ell=1}^{d}(\mathbf{n}_i)_{\ell}(\mathbf{n}_j)_{\ell} &\simeq d\mathbb{E}[\text{product of two independent zero-mean Gaussian random variables}] \\ &= d\left(\mathbb{E}[\text{zero-mean Gaussian random variable}]\right)^2 = 0. \end{aligned} \tag{9}$$

When $i = j$, the approximation

$$\begin{aligned} \mathbf{n}_i^{\top}\mathbf{n}_i = \sum_{\ell=1}^{d}(\mathbf{n}_i)_{\ell}^2 &\simeq d\mathbb{E}\left[\left(\text{zero-mean Gaussian random variable with variance } \sigma^2\right)^2\right] \\ &= d\sigma^2 \end{aligned} \tag{10}$$

holds.

Given Eqs. 9 and 10, the matrix of pairwise correlation among training examples, for task-unrelated dimensions, can be approximately written as

$$N^{\top}N \simeq d\sigma^2\mathbb{I}_n. \tag{11}$$

Eqs. 8 and 11 lead to the result in Eq. 3.

A.2 Task-related/unrelated input dimensions as a non-tight frame and asymptotic regimes

We derive the exact solution in presence of both task-related and task-unrelated input dimension. Let X be defined as in Section 2.1, $N = [\mathbf{n}_1 \cdots \mathbf{n}_n] \in \mathbb{R}^{d(1-\nu)\times n}$ be the matrix of task-unrelated dimensions, and $TX = [T\mathbf{x}_i \cdots T\mathbf{x}_n] \in \mathbb{R}^{d\nu \times n}$ be the task-related dimensions, with $T \in \mathbb{R}^{d\nu\times p}$ generic linear transformation. The input matrix $[X^\top \; N^\top \; (TX)^\top]^\top$ is equivalent to the concatenation of these terms.

Given the presence of redundant rows, we can express the input matrix using the frame formalism as

$$\begin{pmatrix} X \\ N \\ TX \end{pmatrix} = F \begin{pmatrix} X \\ N \end{pmatrix}, \text{ with } F = \begin{pmatrix} \mathbb{I}_p & \mathbf{0}_{p\times d(1-\nu)} \\ \mathbf{0}_{d(1-\nu)\times p} & \mathbb{I}_{d(1-\nu)} \\ T & \mathbf{0}_{d\nu\times d(1-\nu)} \end{pmatrix}, \tag{12}$$

and $\mathbf{0}$ denoting the matrix with null entries and dimensions as specified in the subscripts.

Since we apply the frame on the matrix of independent dimensions, we compute the solution leveraging on Eq. 2. We first evaluate the term $F^\top F$, which assumes the form

$$F^\top F = \left(\begin{array}{c|c} \mathbb{I}_p + T^\top T & \mathbf{0}_{p\times d(1-\nu)} \\ \hline \mathbf{0}_{d(1-\nu)\times p} & \mathbb{I}_{d(1-\nu)} \end{array}\right), \tag{13}$$

and it is invertible. By substituting this term in the exact solution of Eq. 2, we obtain

$$\begin{aligned} W^+ &= Y \left(\left[X^\top \; N^\top\right] F^\top F \begin{bmatrix} X \\ N \end{bmatrix} \right)^{-1} \left[X^\top N^\top\right] F^\top \\ &= Y \left(X^\top \left(\mathbb{I}_p + T^\top T\right) X + N^\top N \right)^{-1} \left[X^\top \left[\mathbb{I}_p \;\; T^\top\right] \;\; N^\top \right]. \end{aligned} \tag{14}$$

This result does not rely on any assumption on the distribution of the task-unrelated dimensions, neither on a specific form of the transformation T.

To get a clearer picture of the effect of redundancy in presence of task-unrelated dimensions, in the following we make three assumptions: (i) the redundant dimensions considered are $r = d\nu/p$, repetitions of every feature in the minimal set; (ii) the vector components for the task-unrelated dimensions are drawn from zero-mean Gaussian distributions sharing the same variance value; and (iii) $d(1-\nu)$ number of task-unrelated dimensions is large.

Given assumption (i), T is a tight frame and the quantity $T^\top T$ corresponds to the identity in $\mathbb{R}^p$, with the scaling factor fixed at r. This implies that the term $\mathbb{I}_p + T^\top T$ is equivalent to $(r+1)\mathbb{I}_p$.

Solution 14, when applied on a new test set $\left[\mathbf{x}_{\text{test}}^\top \; \mathbf{n}_{\text{test}}^\top \; (T\mathbf{x}_{\text{test}})^\top\right]^\top$, corresponds to

$$W^+ \left[\mathbf{x}_{\text{test}}^\top \; \mathbf{n}_{\text{test}}^\top \; (T\mathbf{x}_{\text{test}})^\top\right]^\top = Y \left((r+1)X^\top X + N^\top N\right)^{-1} \left((r+1)X^\top \mathbf{x}_{\text{test}} + N^\top \mathbf{n}_{\text{test}}\right). \tag{15}$$

Under assumptions (ii) and (iii), we leverage on the law of large numbers, as in Appendix A.1. In this case, the sum over the task-unrelated components consists of $d(1-\nu)$ terms, and the approximation $N^\top \mathbf{n}_{\text{test}} \simeq \mathbf{0}$ holds. Using the same argument, the correlation between training examples, when limited to task-unrelated components, approximates a diagonal matrix, as shown in Eqs. 9 and 10. Here the summation is over $d(1-\nu)$ task-unrelated components, leading to $N^\top N \simeq d(1-\nu)\sigma^2 \mathbb{I}_n$.

The prediction for a test sample corresponds to

$$\begin{aligned} W^+ \left[\mathbf{x}_{\text{test}}^\top \; \mathbf{n}_{\text{test}}^\top \; (T\mathbf{x}_{\text{test}})^\top\right]^\top &\simeq Y \left((r+1)X^\top X + d(1-\nu)\sigma^2 \mathbb{I}_n\right)^{-1} X^\top \mathbf{x}_{\text{test}}(r+1) \\ &= Y \left(X^\top X + \frac{d(1-\nu)}{r+1}\sigma^2 \mathbb{I}_n \right)^{-1} X^\top \mathbf{x}_{\text{test}}. \end{aligned} \tag{16}$$

To analyze the interplay between redundant and task-unrelated dimensions, let $X = U\Sigma V^\top$ be the equivalent formulation for the minimal dimensions of the input data, resulting from the singular value decomposition. The terms $U \in \mathbb{R}^{p\times p}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, their columns contain singular vectors in $\mathbb{R}^p$ and $\mathbb{R}^n$ respectively, $\Sigma \in \mathbb{R}^{p\times n}$, matrix of singular values, contains n nonzero values on the diagonal only. Formula 16 is equivalent to

$$W^{+}\left[\mathbf{x}_{\text{test}}^{\top}\ \mathbf{n}_{\text{test}}^{\top}\ (T\mathbf{x}_{\text{test}})^{\top}\right]^{\top} \simeq Y\left(V\left(\Sigma^{\top}\Sigma + \frac{d(1-\nu)}{r+1}\sigma^2\mathbb{I}_n\right)V^{\top}\right)^{-1}X^{\top}\mathbf{x}_{\text{test}}. \tag{17}$$

Using the definition $r = d\nu/p$, we consider the inverse term in Equation 17, with a focus on the input matrix eigenvalues

$$\Sigma^{\top}\Sigma + \frac{pd(1-\nu)}{p+d\nu}\sigma^2\mathbb{I}_n. \tag{18}$$

In the regime of $d \gg p$, the second term in the summation corresponds approximately to $p\sigma^2(1-\nu)/\nu$. The weight on the Tikhonov regularization term depends only on the ratio between number of irrelevant and redundant features.

In the opposite regime, for $p \gg d$, the term $d\nu$ in Expression 18 becomes negligible. The weight of the regularization term is $\sigma^2 d(1-\nu)$.

We finally observe that, as redundancy increases, in the limit of ν tending to 1, the problem is equivalent to the one formulated for the minimal dimensions, with null contribution from the task-unrelated features. This highlights the positive contribution of redundancy on the solution, when restricted to the informative dimensions.

B SYNTHETIC EXPERIMENTS

B.1 REGRESSION PROBLEM WITH CORRUPTED OUTPUT

To analyze a more realistic regression problem, we consider the case of corrupted output. We verify (i) the similarity between unnecessary task-unrelated dimensions and the Tikhonov regularization term, (ii) the benefit of the ℓ_2 regularization, or the presence of unnecessary input dimensions, in presence of corrupted output values.

To this aim, we generate an overparameterized regression dataset, with $p = 10$ minimal dimensions, $o = 4$ output dimensions, and $d = 500$ unnecessary task-unrelated dimensions. The d minimal input dimensions are drawn from $\mathcal{N}(0, \sigma^2_{\text{input}})$, zero-mean Gaussian distribution with a shared variance value. The number of training examples across experiments is $n = 7$. The corrupted output for sample i is

$$\mathbf{y}_i = W^*\mathbf{x}_i + \varepsilon, \tag{19}$$

with zero-mean Gaussian distributed random variable $\varepsilon \sim \mathcal{N}(0, \sigma^2_{\text{output}})$. Across the experiments we increase the corruption level by increasing the variance for the distribution of the variable ε.

We will refer to *w irrelevant* solution, as the the solution computed from the tuple $\left(\left[\mathbf{x}_i^{\top}\ \mathbf{n}_i^{\top}\right]^{\top}, \mathbf{y}_i\right)_{i=1}^{n}$, using Formula 3.

In absence of task-unrelated input dimensions, or for the dataset $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$, we compute the *w/o irrelevant* solution, as in Formula 1, and *Tikhonov* regularized solution, with $\lambda = d\sigma^2_{\text{input}}$ regularization term.

The prediction errors for the three cases correspond respectively to

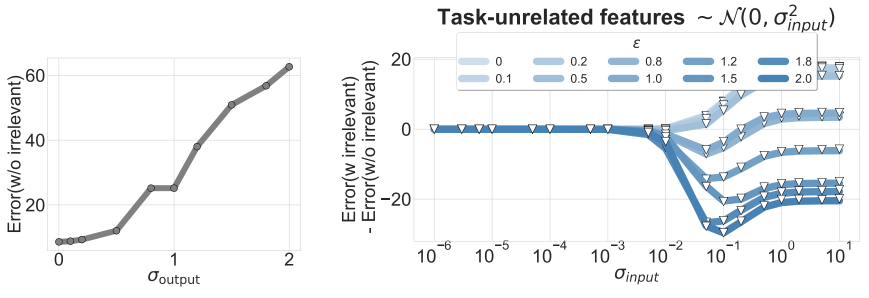
$$\text{Error}(\textit{w irrelevant}) = \left\langle \left\| Y(X^{\top}X + N^{\top}N)^{-1}[X^{\top}\ \ N^{\top}][\mathbf{x}_{\text{test}}^{\top}\ \mathbf{n}_{\text{test}}^{\top}]^{\top} - \mathbf{y}_{\text{test}} \right\|_2^2 \right\rangle, \tag{20}$$

$$\text{Error}(\textit{w/o irrelevant}) = \left\langle \left\| Y(X^{\top}X)^{-1}X^{\top}\mathbf{x}_{\text{test}} - \mathbf{y}_{\text{test}} \right\|_2^2 \right\rangle, \tag{21}$$

$$\text{Error}(\textit{Tikhonov}) = \left\langle \left\| Y(X^{\top}X + \lambda\mathbb{I}_n)^{-1}X^{\top}\mathbf{x}_{\text{test}} - \mathbf{y}_{\text{test}} \right\|_2^2 \right\rangle, \tag{22}$$

where the sample average $\langle\,\cdot\,\rangle$ is computed on the test set examples.

In Fig. 8a, we quantify the difference between Eq. 20 and Eq.21, as the variance of task-unrelated input features increases. Each blue curve is related to a different level of output corruption. For small



(a) (b)

Figure 8: Regression problem with corrupted output: (a) prediction error in absence of unnecessary dimensions, as function of the output corruption; (b) analysis of prediction error as function of the variance of unnecessary components; on the y-axis, difference between test errors in presence (w irrelevant) and absence (w/o irrelevant) of task-unrelated dimensions, a blue curve for each level of output corruption. (b)

values of σ_{input}, the presence of task-unrelated dimensions does not have any effect on the test error. As σ_{input} increases, irrelevant dimensions harm the solution for very small output corruption, while gradually benefiting the prediction as the output corruption increases.

The white triangles in Fig. 8a correspond to the difference between Eq. 20 and Eq. 22. These points follow each of the blue curves, showing (i) the valid approximation of term 20 with Eq. 22 and, consequently, the regularization effect given by additional task-unrelated input dimensions. In the limit of large d, the presence of additional Gaussian features with zero mean acts as a Tikhonov regularization term, with ℓ_2 constraint on the norm.

Lastly, in Fig. 8b, we report the test error computed as in Eq. 21. As the labels corruption increases, we observe consistently higher test errors.

These results highlight the controversial effect of additional task-unrelated dimensions, which can effectively benefit the prediction performance in presence of corrupted output.

B.2 Optimization protocol

MLPs trained on synthetic teacher classification datasets share the same optimization protocol. We fix the maximum number of epochs at 100. The convergence criterion is based on early stopping, with tolerance value 10^{-4} on the validation loss and patience of 8 epochs. Optimal batch size and learning rate are selected through a grid search procedure, respectively over the arrays $[2, 10, 32, 50]$ and $[10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$. We eliminate elements of the batch size array from the hyper-parameters search if those are larger or equal to the number of training examples. We apply a reduction factor $1/10$ on the learning rate as the validation loss shows over five epochs, variations smaller than 10^{-4}. The weights initialization adopted across networks and layers is the Glorot uniform, *ie.* uniform distribution in the interval $[-u, u]$, where $u = \sqrt{6/(\text{fan}_{\text{in}} + \text{fan}_{\text{out}})}$, with fan_{in} and fan_{out} respectively number of input and output units.

We report mean and standard deviation of the AUTC values across three repetitions of each experiment.

B.3 Further experiments on data separability

We test the generality of our result as we decrease the distance between the distributions of the two classes. We preserve the hypothesis of standard distribution for task-unrelated components. We generate a sample $\mathbf{x}$ from class c as $\mathbf{x} \sim (\mathcal{N}(\mu_1^c, (\sigma_1^c)^2), \ldots, \mathcal{N}(\mu_p^c, (\sigma_p^c)^2))$. The values μ_ℓ^c, with c denoting the class, and ℓ one among the minimal dimensions, are random values. We report in the following the distance between the two classes, computed as $D = \sqrt{\sum_{\ell=1}^{p}(\mu_\ell^0 - \mu_\ell^1)^2}$. At each repetition we compute this distance over the resulting training set (from which we extract n_{tr} examples) of 10000 examples and we obtain the following distance mean values $\langle D \rangle$ across 3

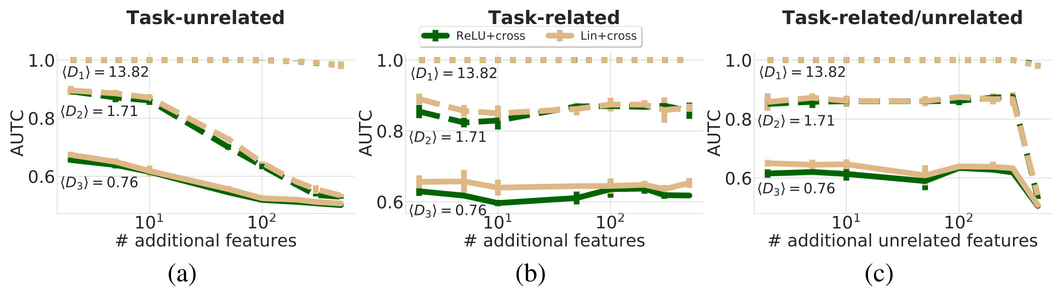


Figure 9: AUTC curves for the multivariate Gaussian distributions, as we decrease the distance D between two classes: (a) task-unrelated, (b) task-related, (c) task-related/unrelated cases.

repetitions: $\langle D_1 \rangle = 13.82 \pm 0.84$, $\langle D_2 \rangle = 1.71 \pm 0.04$, and $\langle D_3 \rangle = 0.76 \pm 0.01$. In Figure 9 we report the AUTC values for MLPs with ReLU non-linearity and linear activation and cross entropy loss, evaluated on $n_{tr} = \{2, 5, 10, 20, 50\}$ examples per classes.We generate additional task-unrelated dimensions using the standard $\mathcal{N}(0, 1)$ distribution. In Fig.9(a), the MLPs are affected by the presence of additional dimensions, with exception of the most separable datasets, with $\langle D_1 \rangle = 13.82$. This result must be compared to the one in Fig.9(b), for task-related dimensions. Here the performance does not depend on the amount of unnecessary dimensions. In Fig.9(c) we report the AUTC as we fix the amount of unnecessary dimensions to 500. We increase the ratio between task-unrelated over redundant dimensions, from the left to the right of the horizontal axis. In (c), as for previous experiments, the data efficiency is higher for case in (c) than for case in (a).

B.4 Area Under the Test Curve (AUTC)

We use the an area under the curve to report the data efficiency of a network. In particular, we define the Area Under the Test Curve (AUTC) as a measure of the generalization performance for different training set size, as defined in Eq. 23

$$\text{AUTC}(d) = \frac{\text{AUC}(n_{tr}, \text{test accuracy}(n_{tr}, d))}{\max(n_{tr}) - \min(n_{tr})}, \tag{23}$$

where d denotes the number of dimensions, n_{tr} the amount of examples in the training set and we use the implementation of the AUC by scikit-learn[1].

The AUTC measure has value in the interval $[0, 1]$. The denominator has value equivalent to those of a rectangle of height 1 or the most data efficient model. Depending on the experiments, if the array of n_{tr} varies of orders of magnitudes, we may be interested in giving the same emphasis to small as large n_{tr} values. For these cases, we define a further measure, the log AUTC, where the n_{tr} value is mapped through a $\log_{10}$, as in Eq. 24

$$\log \text{AUTC}(d) = \frac{\text{AUC}(\log_{10}(n_{tr}), \text{test accuracy}(n_{tr}, d))}{\max(\log_{10}(n_{tr})) - \min(\log_{10}(n_{tr}))}. \tag{24}$$

C Deep convolutional neural networks

C.1 Synthetic MNIST data generation

Fig. 10 shows a larger version of the three Synthetic MNIST datasets. In all the cases, the normalize the MNIST digits by 255. For the task-unrelated case, in (a), the image size increases as we add random pixels at the edge, which constitute unrelated dimensions. These are the absolute value of numbers extracted from a zero-mean Gaussian distribution with $\sigma = 0.2$. The MNIST edge size has size 28 across all the task-unrelated versions for this dataset. For the task-related case, in (b), we upscale the MNIST digit using bicubic interpolation. For the task-related + unrelated case, in (c), we fix the image size to 200×200. From the left to the right we increase the task-related dimensions by

[1]`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html` (Last access: Oct. 1, 2020)

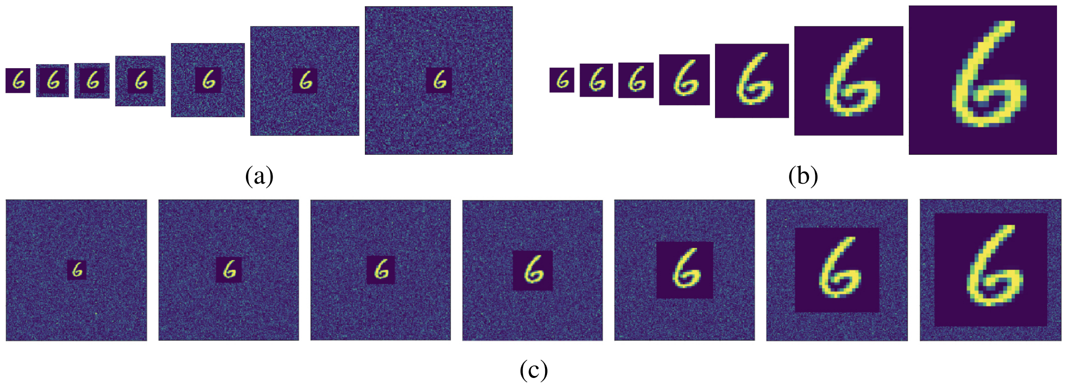


Figure 10: An example from each version of the Synthetic MNIST dataset. (a) task-unrelated case, (b) task-related case, (c) task-related/unrelated case, the image size is fixed at 200.

upscaling the MNIST object. In (a) and (b) the images size are $k = \{28, 36, 40, 56, 80, 120, 160\}$. For case (c), the total image size is fixed at 200, while the upscaled MNIST edge has values $k = \{28, 36, 40, 56, 80, 120, 160\}$.

C.2 Details about DCNN architectures for Synthetic/Natural MNIST

We leverage on the MLP with ReLU activations and cross entropy loss, with identical hyperparameters of those in Section 2.3. The MLP is provided with a flatten version of the image as input.The DCNNs consist of three convolutional layers (iC-iiiC) followed by a flatten operation and two fully connected layers (iL and iiL). In Tab. 2 we report: at the first row the number of filters or nodes, depending on the layer; at the second row, the size of filters and max-pooling operations; and at the last row, the non-linearity used, in any. The filters and max-pool operations have square dimensions, and their sizes at the first layers vary across experiments.

Table 2: DCNN for image classification. From the top: number of filters *C or nodes *L at each hidden layer, filters size and max pooling sizes for *C; activation functions following *C or *L.

	iC	iiC	iiiC	iL	iiL
#filters/#nodes	32	64	64	64	10
filters, max-pooling	var, var	3, 2	3, 2	none	none
function	ReLU	ReLU	ReLU	ReLU	soft-max

C.3 Optimization protocol for Synthetic MNIST

MLP and DCNN share the same optimization protocol. Models are optimized using stochastic gradient descent with null momentum. Batch size and learning rate are selected using a grid search procedure. The learning rate array has values $[1, 8 \cdot 10^{-1}, 5 \cdot 10^{-1}, 2 \cdot 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 5 \cdot 10^{-6}, 10^{-6}]$, the batch size array is $[10, 32, 50, 100]$. We eliminate elements from the batch size array if those exceed the training set size. The convergence criterion is based on early stopping, convergence is reached when the validation loss shows across 10 repetitions variations smaller than Δ, with $\Delta = 10^{-6}$. The maximum amount of epochs is fixed at 500. We reduce the learning rate by a factor $1/10$ when variations of the validation loss are smaller than 10^{-4} across 5 epochs.

The initialization weight values across architectures is based on the Glorot uniform distribution, or uniform distribution in the interval $[-u, u]$, where $u = \sqrt{6/(\text{fan}_{\text{in}} + \text{fan}_{\text{out}})}$, with fan_{in} and fan_{out} respectively number of input and output units.

C.4 Optimization protocol for Natural MNIST

All the networks share the same optimization protocol. We implement a stochastic gradient descent optimizer with null momentum. We chose the best batch size and learning rate through a grid search procedure. The learning rate grid η has values $[5 \cdot 10^{-1}, 2 \cdot 10^{-1}, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$, the batch size array has values in $[10, 32, 50, 100]$. We fixed the maximum amount of epochs at 500. Convergence criterion and learning rate reduction are as defined in Section C.3. We initialized the networks using the Glorot uniform distribution, or uniform distribution in the interval $[-u, u]$, where $u = \sqrt{6/(\text{fan}_{\text{in}} + \text{fan}_{\text{out}})}$, with fan_{in} and fan_{out} respectively number of input and output units.

C.5 Performance for large n_{tr} and activations on Natural MNIST

In Fig. 11, the networks generalize with almost perfect accuracy for $n_{tr} = 1000$. The dataset considered has the largest amount of unnecessary dimensions. We compare the networks hidden

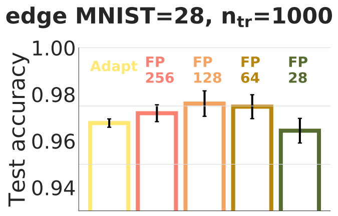


Figure 11: Test accuracies on Natural MNIST with edge MNIST / edge image 11%, trained on $n_{tr} = 1000$.

representations as n_{tr} decreases. We provide here further details about the generation of Fig. 6(b) and (c). We compute activations, or representations at each hidden layer after non-linearity, using a subsample of 1000 points from the test set. For each image, at a fixed layer, we compute the maximum representation value (or the maximum activation), across spatial dimensions and filters. We normalize the representation at this layer by using this value. This operation reduces all the activations to have values between $[0, 1]$. We divide the representation in five regions, from the edge to the center, and we store the maximum normalized value for each region. We report the distribution of those values across the test samples. We expect efficient networks to be highly selective for the image center. In the ideal case, the distribution for the activation values at the center should be peaked at one. The distributions of activations values should be shifted to lower averages as we move to regions distant from the center.

C.6 The effect of global average pooling on the generalization performance

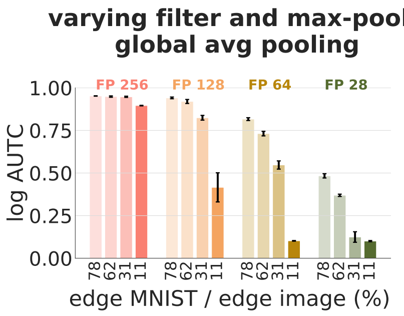


Figure 12: log AUTC on Natural MNIST. All the networks have an average global pooling operation after the three convolutional layers.

Fig. 12 shows the log AUTC values for networks trained on Natural MNIST. After the last convolutional layer we introduce an average pooling operation across the spatial dimensions. The two MLP

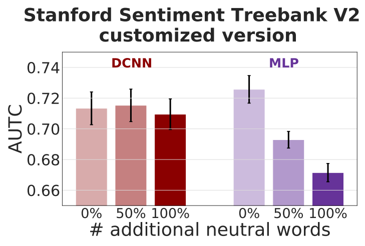


Figure 13: AUTC on SST2 dataset, splitted on training set.

layers share the same amount of free parameters across all the DCNNs. For FP 256, the performance is close to a perfect score across different dataset versions. FP 28 and DCNNs with smaller filters and max-pooling sizes instead heavily suffer from the presence of the background. The convolutional layers alone are not sufficient to discard the background and reach good generalization performance, despite the reduction of free parameters caused by the global pooling operation.

D STANFORD DOGS

All the images from the original dataset, for all the cases shown in Fig. 7, are resized to dimensions 227×227, the input dimensions we fix for our ResNet18 model. We subtract the mean values for the ImageNet dataset to center the RGB channels, with $R_{\text{mean}} = 123.68$, $G_{\text{mean}} = 116.78$, and $B_{\text{mean}} = 103.94$. We split the original training set into a training and a validation set, the former consisting of 90 examples per class (breed), the latter consisting of 10 examples per class. To measure the AUTC, we varied n_{tr}, corresponding to the amount of training examples. Across all the experiments, the learning rate is $\eta = 0.128$ at the first iteration. We then divide it by a factor of 10 every time we reach a plateau of the validation accuracy. All the models are trained until we reach the plateau for the smallest learning rate considered, $\eta = 1.28 \cdot 10^{-3}$. In all cases we observed no improvement of the validation accuracy after the plateau at $\eta = 1.28 \cdot 10^{-2}$.

E SENTIMENT ANALYSIS

We leverage on Stanford Sentiment Treebank v2 (Socher et al., 2013), SST2 in short, a popular dataset of rating movie reviews for text classification[2]. The learning and test splits contain labeled sentences, from 0 to 4, which can be reduced to a binary classification problem (0-1 negative reviews, 2 neutral reviews excluded from the binary dataset, 3-4 positive reviews). The training set contains a further set of phrases; chunks of sentences with corresponding labels. Controlling the amount of task-related dimensions is hard, because of the possible presence of neutral words in positive/negative phrases, while we assume that neutral phrases contain neutral words only, or intuitively task-unrelated quantities. We manipulate their amount by collecting, for each sentence, all the available labeled phrases that are contained in the sentence. In analogy to the definition of unnecessary task-unrelated dimensions of Section 2.1, this corresponds to the repetition of the least informative components. We expect the generalization performance to decrease as we add more of these dimensions to the input.

Our training, validation, and test sets are generated from the original SST2 training set. These splits consist respectively of 3800, 872, and 1821 balanced examples. The percentage of unnecessary task-unrelated input phrases takes values $[0, 1/2, 1]$, with 0 corresponding to no additional phrases, and 1 corresponding to all the possible neutral phrases contained in the sentence. We rely on the word2vec embedding (Mikolov et al., 2013). We compare the MLP ReLU + cross entropy model, trained on the average embedding for each sentence, to a standard DCNNs for text classification (Kim, 2014), with filters of sizes (3,4,5), each followed by a global max-pooling operation. MLP and DCNN share the same optimization protocol: AdaDelta optimizer with decay rate $\rho = 0.95$ and constant $\epsilon = 10^{-7}$ fixed. We perform a grid search over the learning rate and the batch size

[2]We follow some standard preprocessing guidelines. Dataset and preprocessing (de-capitalization, punctuations removal) as in `https://github.com/CS287/HW1/tree/master/data` (Last access: Oct 1, 2020).

parameters. The learning rate array has values $\eta = [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 2, 5]$ while the batch size array has values $[5, 50]$, when the number of training examples is larger than 50, while it is fixed to 5 otherwise. The convergence criterion is based on early stopping and it is reached when the validation loss shows smaller variations than Δ, with $\Delta = 10^{-4}$, across 8 epochs. We fixed the maximum amount of epochs at 100. We apply a reduction factor of $1/10$ on the learning rate anytime that the validation loss shows smaller variation than 10^{-4} in 8 epochs. Across architectures we initialized networks weights with the Glorot uniform distribution.

For the DCNN architecture, as in (Kim, 2014), we fix at 3 the ℓ_2 regularization with bound on the norm, soft-max at the last layer and cross-entropy cost function. We run 10 repetitions of each experiment for different number of training examples per class, in the range $[5, 250]$.

Fig. 13 shows the extreme robustness of the DCNN to the presence of unnecessary task-unrelated features, differently from the MLP case. In contrast with the synthetic case, the gap in generalization performance for a different amount of unnecessary input features does not reduce as we increase the amount of training samples. In our opinion, this is due to the global average pooling on the embedding.