Adaptive structured noise injection for shallow and deep neural networks

Beyrem Khalfaoui, Joseph Boyd, Jean-Philippe Vert

Adaptive structured noise injection for shallow and deep neural networks

Beyrem Khalfaoui [1,2] , Joseph Boyd [1,2], Jean-Philippe Vert [3,1]

[1] MINES ParisTech, PSL Research University,
CBIO - Centre for Computational Biology, F-75006 Paris, France
[2] Institut Curie, PSL Research University, INSERM, U900, F-75005 Paris, France.
[3] Google Brain, F-75009 Paris, France.
beyrem.khalfaoui@mines-paristech.fr, jpvert@google.com

Abstract

Dropout is a regularisation technique in neural network training where unit activations are randomly set to zero with a given probability *independently*. In this work, we propose a generalisation of dropout and other multiplicative noise injection schemes for shallow and deep neural networks, where the random noise applied to different units is not independent but follows a joint distribution that is either fixed or estimated during training. We provide theoretical insights on why such adaptive structured noise injection (ASNI) may be relevant, and empirically confirm that it helps boost the accuracy of simple feedforward and convolutional neural networks, disentangles the hidden layer representations, and leads to sparser representations. Our proposed method is a straightforward modification of the classical dropout and does not require additional computational overhead.

1 Introduction

The tremendous empirical success of deep neural networks (DNN) for many machine learning tasks such as image classification and object recognition (Krizhevsky et al., 2017) contrasts with their relatively poor theoretical understanding. One feature commonly attributed to DNN to explain their performance is their ability to build hierarchical *representations* of the data, able to capture relevant information in the data at different scales (Bengio et al., 2013; Tishby and Zaslavsky, 2015; Mallat, 2012). An important idea to create good sets of representations is to reduce redundancy and increase diversity in the representation, an idea that can be traced back to early investigations about learning (BARLOW, 1959) and that has been implemented in a variety of methods such as independent component analysis (Hyvärinen, 2013) or feature selection (Peng et al., 2005). Explicitly encouraging diversity has been shown to improve the performance of ensemble learning models (Kuncheva and Whitaker, 2003; Dietterich, 2000), and techniques have been proposed to limit redundancy in DNN by pruning units or connections (Hassibi and Stork, 1993; LeCun et al., 1990; Mariet and Sra, 2016) or by explicitly encouraging diversity between units of each layer during training (Cogswell et al., 2015; Desjardins et al., 2015; Rodríguez et al., 2016; Luo, 2017).

Dropout (Hinton et al., 2012; Srivastava et al., 2014) is a recent and popular regularisation techniques in deep learning that exploits the idea of creating diversity stochastically while training, by randomly setting to zero come units or connections during stochastic gradient optimization. Proposed by Hinton et al. (2012) as a way to prevent co-adaptation of units and to approximately combine exponentially many different DNN architectures efficiently, it has improved DNN performance in many benchmark datasets. Dropout can also be interpreted as a regularization technique (Baldi and Sadowski, 2013; Wager et al., 2013; Maeda, 2014; Helmbold and Long, 2017), however its impact on learning a good representation of the data remains elusive. Several variants of the original dropout model have been proposed to adapt the algorithm to other models (e.g., Gal and Ghahramani, 2016), or to modify the distribution of the stochastic noise. Ba and Frey

(2013) propose that the dropout rate of a unit should depend on its magnitude and dynamically adapt the dropout rate of each unit activation during training. Another variant is to group units together because of their proximity in a map (Tompson et al., 2014; DeVries and Taylor, 2017) of because they are strongly correlated (Aydore et al., 2018), before applying dropout jointly on the units in a group. This can equivalently be interpreted as applying dropout to individual units, but constraining the stochastic dropout noise in units within a group to be perfectly correlated.

In this work, we extend and generalize the idea to modify the noise distribution, and analyse both theoretically and empirically the effect of different choices. In particular, we study the impact of creating correlations among noise in the units of a given layer, generalizing the ideas of Tompson et al. (2014); DeVries and Taylor (2017); Aydore et al. (2018) to a general covariance structure. We depart from binary dropout noise to the more flexible multiplicative Gaussian noise model, which allows to specify any covariance structure without the need to explicitly cluster units into groups, and highlight the role of the noise correlation matrix in the regularization effect of this structured noise injection procedure. We show in particular that borrowing the covariance of the units to create the covariance of the noise can decrease redundancy among the units, a phenomenon we confirm empirically that leads to better representations and classification accuracy.

2 Dropout and multiplicative noise

Let us first set notations to describe a standard neural network for inputs in $\mathbb{R}^d$ with H layers of respective dimensions $d_1, \dots, d_H$. For any layer $l \in [1, H]$ let $W^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ and $b^{(l)} \in \mathbb{R}^{d^{(l)}}$ denote respectively the matrix of weights and the vector of biases at layer l, with the convention $d^{(0)} = d$, and let $\theta = (W_l, b_l)_{l=1,\dots,H}$ denote the set of parameters of the network. The network defines a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^{d^{(H)}}$ given for any $x \in \mathbb{R}^d$ by $f_\theta(x) = y^{(H)}$, where $y^{(l)}$ is defined recursively for $l = 0, \dots, H$ by $y^{(0)} = x$ and, for $l \in [1, H]$:

$$\begin{cases} z^{(l)} &= W^{(l)} y^{(l-1)} + b^{(l)} \,, \\ y^{(l)} &= \sigma^{(l)}(z^{(l)}) \,, \end{cases}$$

where $\sigma^{(l)}$ is an activation function at the l-th layer, such as the RELU function $\sigma(t) = t\mathbf{1}(t > 0)$ for $t \in \mathbb{R}$, applied entrywise if its input is a vector.

Given a training set $\mathcal{D}$ of N labelled inputs $(x_1, y_1), \dots, (x_N, y_N) \in \mathbb{R}^d \times \mathbb{R}^p$, and a loss function $L : \mathbb{R}^{d^{(H)}} \times \mathbb{R}^p \to \mathbb{R}$, training the neural network amounts to fitting the parameters θ by trying to minimize the average loss over the training set:

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} L\left(f_\theta\left(x_i\right), y_i\right) \,, \tag{1}$$

usually by some form of stochastic gradient descent (SGD) using backpropagation to compute gradients. For example, when the label is a scalar ($p = 1$), then one can use $d^{(H)} = 1$ and the squared error $L(u, v) = (u - v)^2$ for $u, v \in \mathbb{R}$.

A popular way to improve the training of neural networks is to use dropout regularisation, where during training the units of the input and hidden layers are stochastically omitted during the back-propagation algorithm (Srivastava et al., 2014). Dropout is a particular case of multiplicative noise, which we can formalize as follows. Given a sequence of vectors $r = \left(r^{(0)}, \dots, r^{(H-1)}\right) \in \mathbb{R}^{d^{(0)}} \times \dots \times \mathbb{R}^{d^{(H-1)}}$, of total dimensions $D = d^{(0)} + \dots + d^{(H-1)}$, we create the modified function $f_\theta(x, r) = y^{(H)}$ where $y^{(0)} = x$ and, for $l \in [1, H]$:

$$\begin{cases} \widetilde{y}^{(l-1)} &= r^{(l-1)} \odot y^{(l-1)} \,, \\ z^{(l)} &= W^{(l)} \widetilde{y}^{(l-1)} + b^{(l)} \,, \\ y^{(l)} &= \sigma^{(l)}(z^{(l)}) \,, \end{cases}$$

where $\odot$ denotes the Hadamard or element-wise product. We then define a set of independent and identically distributed (i.i.d.) random variables $(R_i)_{i=1,\dots,N}$ with values in $\mathbb{R}^D$ (the "noise") and train the network by solving

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathbb{E} L\left(f_\theta\left(x_i, R_i\right), y_i\right) \,. \tag{2}$$

With these notations, the standard dropout approach of (Srivastava et al., 2014) with parameter $p \in [0, 1]$ is obtained by taking a noise distribution with i.i.d. entries across the dimensions taking the value $1/p$ with probability p, and 0 with probability $1 - p$.

3 Structured noise injection (SNI)

We propose to create new learning schemes by learning with multiplicative noise, as described above, where the noise distribution of R is *not* i.i.d across the units (while we keep the noise samples $R_1, \ldots, R_N$ i.i.d. according to the noise distribution). For simplicity, we focus only on Gaussian noise:

$$R \sim \mathcal{N}(\mathbf{1}_D, \lambda\Sigma)\,, \tag{3}$$

where $\mathbf{1}_D$ is the constant D-dimensional vector with all entries equal to 1, $\lambda \geq 0$ is a regularization parameter and $\Sigma \in \mathbb{R}^{D\times D}$ is a symmetric positive semidefinite covariance matrix. When $\lambda = 0$, R is almost surely constant equal to $\mathbf{1}$, and we recover the standard learning without noise injection (1). When $\lambda > 0$ and $\Sigma = \mathbf{I}$ we learn from i.i.d. multiplicative Gaussian noise, a variant of dropout that was proposed by Srivastava et al. (2014) and shown to perform very similarly to dropout. When Σ is diagonal but not necessarily constant on the diagonal, the amount of noise can vary among units, but the noise is still independent across units.

Our focus in this paper is on non-diagonal covariance matrices Σ, which create correlations between the noise at different units. We call this setting *structured noise injection (SNI)*. In the case of multilayer neural network, it is natural to create correlations within layers, and not between layers, which translates to a block-diagonal structure for Σ, where each block corresponds to the units of a given layer. We further consider two flavors of SNI.

3.1 SNI with fixed noise covariance

The basic flavor of SNI is when we fix the noise covariance Σ *a priori* and independently from the data, using for example the structure of the network as prior knowledge. This is a way to input prior knowledge about the problem in the learning algorithm, and has already been proposed as a promising technique in different settings, particularly with binary noise. For example, Tompson et al. (2014) proposed the SpatialDropout method, where entire feature maps corresponding to adjacent pixels are randomly discarded together instead of individual pixels, corresponding to binary SNI with a block-diagonal covariance matrix with constant blocks equal to 1 for each set of pixels in a feature map. Similarly, DeVries and Taylor (2017) applies binary SNI at the input layer of a convolutional network by masking contiguous sections of inputs rather than individual pixels, yielding new state-of-the-art results in image classification. Implementing a SNI strategy with fixed, block-diagonal covariance matrix at the layer level, is only a slight generalization of standard parameter inference with SGD and backpropagation. For example, Algorithm 1 illustrates the forward pass between layers $l-1$ and l with SNI regularization, where $\Sigma^{(l-1)}$ is the block of Σ corresponding to layer $l-1$.

Algorithm 1 Feed-forward pass with SNI at layer l

Input: Mini-batch of outputs from the previous $(l-1)$-th layer $y_1^{(l-1)}, \ldots, y_n^{(l-1)} \in \mathbb{R}^{d^{(l-1)}}$, regularization parameter $\lambda \in \mathbb{R}_+$, covariance matrix of the noise $\Sigma^{(l-1)}$

Output: The mini-batch of outputs from the l-th layer

for $i = 1$ to n **do**
  Sample $r_i^{(l-1)} \sim \mathcal{N}(\mathbf{1}_{d^{(l-1)}}, \lambda\Sigma^{(l-1)})$
  $\widetilde{y}_i^{(l-1)} \leftarrow r_i^{(l-1)} \odot y_i^{(l-1)}$
  $z_i^{(l)} \leftarrow W^{(l)}\widetilde{y}_i^{(l-1)} + b^{(l)}$
  $y_i^{(l)} \leftarrow \sigma(z_i^{(l)})$
end for
return $y_1^{(l)}, \ldots, y_n^{(l)} \in \mathbb{R}^{d^{(l)}}$

3.2 Adaptive SNI

Another SNI approach is to define a noise structure with covariance $\Sigma(\mathcal{D}, \theta)$ which may depend on the data and on the model parameters. We refer to this situation as *adaptive structured noise injection (ASNI)*. An example of ASNI approach, where Σ depends on the data $\mathcal{D}$ but not on the model parameters θ, was recently proposed by Aydore et al. (2018): they first use the data $\mathcal{D}$ to identify groups of correlated features, and then perform dropout at the group level, reporting promising empirical results. In other words, they impose a block diagonal correlation structure on the noise, where each block corresponds to a group of correlated features, and the correlation of the noise within a group is 1. While grouping is needed when one wants to perform dropout by group, more general correlation matrices are possible when the noise in Gaussian. An obvious extension of the work of Aydore et al. (2018) is to impose over the noise the same covariance structure as observed in the data. When a single-layer linear model is considered, as in Aydore et al. (2018), then Σ depends only on the data $\mathcal{D}$, but not on the model parameters θ. In the case of multi-layer networks, the covariance of the units at a given internal layer not only depends on the data distribution, but also on the parameters θ of the models; as a result, the covariance of the noise in internal layers depends on both $\mathcal{D}$ and θ when we impose that is equals that of the units in the layer.

To solve (2) with ASNI, we can still follow a SGD approach where at each step a minibatch of examples is randomly chosen, and a realization of the noise on these examples is sampled. One difficulty in ASNI is that the law of the noise depends on the data themselves, and on the parameters of the DNN which evolve during optimization. Similarly to the procedure used in batch normalization (Ioffe and Szegedy, 2015), we propose to re-estimate the covariance of the noise at each SGD iteration from the mini-batch itself, before sampling the noise on the mini-batch using this structure. Algorithm 1 details the feed-forward pass for one layer, in the case where Σ is block diagonal with a block for each layer equal to the covariance of the units in that layer.

We note that in order to sample the noise with a given covariance matrix Σ, one typically needs to factorize $\Sigma = UU^\top$ by spectral decomposition or Cholesky decomposition (Gentle, 2009), and then get the samples as $U\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{1}, \mathbf{I})$. This can create significant computational burden, since this must be performed at each SGD iteration, and is one of the reasons why, for example, batch normalization only scales each feature but does not perform whitening (Ioffe and Szegedy, 2015). In the particular case of ASNI where Σ is the covariance of the data, we can get an important speed-up since the estimate of Σ on a mini-batch is already factorized as $\hat{\Sigma} = \tilde{Y}^\top \tilde{Y}$, where $\tilde{Y}$ is the $n \times d^{(l-1)}$ matrix of mini-batch outputs, centered and divided by $\sqrt{n}$. In other words, the estimation of $\Sigma^{(l-1)}$ in line 1 of Algorithm 2 can be completely bypassed, and the matrix U in line 4 replaced by $\tilde{Y}$. We note that this is particularly efficient when the mini-batch size n is not larger than the number of units. Finally, a further important speed-up is possible by sampling a single noise vector for each sample in a mini-batch, instead of sampling a different vector for each sample. In other words, we can move lines 3 and 4 of Algorithm 2 outside of the "for" loop, between lines 1 and 2. This may come at a price of more iteration, since it increases the variance of the stochastic gradient, but we found in our experiments that it did not significantly affect the number of iterations needed and brought a significant overall speed-up.

Algorithm 2 Feed-forward pass with ASNI at layer l

Input: Mini-batch of outputs from the previous layer $y_1^{(l-1)}, \ldots, y_n^{(l-1)} \in \mathbb{R}^{d^{(l-1)}}$, regularization parameter $\lambda \in \mathbb{R}_+$

Output: The mini-batch of outputs from the l-th layer

1: Estimate $\Sigma^{(l-1)} = UU^\top$ with $U \in \mathbb{R}^{d^{(l-1)} \times d_U}$ from the batch
2: **for** $i = 1$ to n **do**
3: Sample $\epsilon_i \sim \mathcal{N}(\mathbf{1}_{d_u}, \mathbf{I})$
4: $r_i^{(l-1)} \leftarrow \sqrt{\lambda} U \epsilon_i$
5: $\widetilde{y}_i^{(l-1)} \leftarrow r_i^{(l-1)} \odot y_i^{(l-1)}$
6: $z_i^{(l)} \leftarrow W^{(l)} \widetilde{y}_i^{(l-1)} + b^{(l)}$
7: $y_i^{(l)} \leftarrow \sigma(z_i^{(l)})$
8: **end for**
9: **return** $y_1^{(l)}, \ldots, y_n^{(l)} \in \mathbb{R}^{d^{(l)}}$

4 Regularization effect

It is well known that learning with noisy data can be related to regularization. For example, adding uncorrelated Gaussian noise to data in ordinary least squares regression is equivalent, in the case of squared error, to ridge regression on the original data Bishop (1995). The following result clarifies how correlations in the noise impacts this regularization. We consider a setting with no hidden layer and no bias term, i.e., a simple linear model of the form $f(x) = w^\top x$ where $w \in \mathbb{R}^d$ is the vector of weights of the model.

Lemma 1. *Given a training set $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ with centred inputs ($\sum_{i=1}^n x_i = 0$), and given $R_1, \dots, R_N$ i.i.d. random variables following $\mathcal{N}(\mu, \lambda\Sigma)$ for some $\lambda \geq 0$ and covariance matrix Σ, the following holds for any $w \in \mathbb{R}^d$:*

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\left(w^\top (R_i \odot x_i) - y_i\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(w^\top x_i - y_i\right)^2 + \lambda w^\top (C \odot \Sigma)\, w\,,$$

where $C = \frac{1}{n}\sum_{i=1}^N x_i x_i^\top$ is the covariance matrix of the data. Furthermore, if $L : \mathbb{R}^2 \to \mathbb{R}$ is a general loss function and for any $y \in \mathbb{R}$, the function $u \in \mathbb{R} \mapsto \ell_y(u) = L(u, y)$ twice-differentiable, then the following holds:

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}L\left(w^\top (R_i \odot x_i), y_i\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(w^\top x_i, y_i\right) + \frac{\lambda}{2} w^\top \left(J(w) \odot \Sigma\right) w + o(\lambda)\,,$$

where

$$J(w) = \frac{1}{N}\sum_{i=1}^{N} \ell''_{y_i}\left(w^\top x_i\right) x_i x_i^\top\,.$$

The proof of these results follows standard arguments (e.g. Wager et al., 2013). They show how the structure in the noise can be interpreted as a particular regularization. Depending on the noise covariance Σ, we derive several interesting situations:

- In the case of standard dropout regularization with i.i.d. nodes ($\Sigma = \mathbf{I}$), we recover known results of Wager et al. (2013); Baldi and Sadowski (2013).
- When $\Sigma = \mathbf{1}_d \mathbf{1}_d^\top$, i.e., when the noise is the same for all units, then Σ is the neutral multiplication of the Hadamard product. This implies that the regularization boils down to $\lambda w^\top C w$ in the least squares regression case, and to $\lambda w^\top J(x) w/2$ in the more general case. Interestingly, in the least squares regression with centered data, we have the following:

  Lemma 2. *Given a training set $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ with centred inputs ($\sum_{i=1}^n x_i = 0$), and given $R_1, \dots, R_N$ i.i.d. random variables following $\mathcal{N}(\mu, \lambda \mathbf{1}_d \mathbf{1}_d^\top)$ for some $\lambda \geq 0$, the following holds for any $w \in \mathbb{R}^d$:*

  $$\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\left(w^\top (R_i \odot x_i) - y_i\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\tilde{w}^\top \tilde{x}_i - y_i\right)^2 + \lambda \tilde{w}^\top \tilde{w}\,,$$

  where $C = \frac{1}{n}\sum_{i=1}^N x_i x_i^\top$ is the covariance matrix of the data, $\tilde{x}_i$ is a whitened version of x_i for $i = 1, \dots, n$, and $\tilde{w}$ is the whitened version of w.

  In other words, SNI on centered data is equivalent in the case of squared error to standard ridge regression on the whitened data. Remember that whitening data is obtained by multiplying each vector by a whitening matrix $Z \in \mathbb{R}^{d \times d}$ that satisfy $Z^\top Z = C^{-1}$. There exist an infinite number of possible whitening matrix, a standard choice being $Z = C^{-1/2}$ for ZCA whitening. Hence, in Lemma 2, $\tilde{x}_i = Z x_i$ and $\tilde{w} = Z w$, and the proof of Lemma 2 results from simple algebric manipulations of the results of Lemma 1. Interestingly, when noise injection is done per layer, then Lemma 2 shows that injecting the same noise to all units of a given layer is equivalent to data whitening at the layer input, combined with ridge regularization (a.k.a. weight decay in the neural network terminology). Data whitening is usually associated with high computational cost associated to diagonalization of the covariance matrix,

and is replaced in practice by batch normalization which only normalizes the variance of each unit; our analysis suggests that SNI with strong noise correlation within a layer provides a computationally efficient approach to obtain the same result as complete data whitening.

- Finally, we propose $\Sigma = C$ as another interesting structure for ASNI, which generalizes the idea of Aydore et al. (2018) to creating noise correlation on correlated units and in case of non-linear models. In the case of least square regression, this is equivalent by Lemma 1 to a regularization by $w^\top C^{\odot 2} w$. While detailed analysis of this choice is complicated by the fact that the eigenstructure of $C^{\odot 2}$ is not easily related to the one of C, we show empirically below that it leads to promising results.

5 Effect on learned representation

While Lemma 1 interprets SNI as regularization in the one-layer, linear model case, the same analysis can be done at each layer of a multi-layer network. In that case, though, both the inputs of the layer (i.e., the x_i's in Lemma 1) and the weights w are jointly optimized, since the inputs depend on the parameters of the previous layers. Hence SNI may affect the *representation* learned by a multi-layer network.

Let us take the case of least squares regression as an example, where SNI is equivalent to a regularization by $\lambda w^\top (C \odot \Sigma) w$ according to Lemma 1. When $\Sigma = \mathbf{1}_d \mathbf{1}_d^\top$, we have seen that SNI is equivalent to ridge regression on the whitened data. Hence, any rotation of the inputs has no impact on the model learned, since both the whitening of the x_i's and the ridge penalty on w are invariant by rotation. As a consequence, SNI in that case does not promote independence of the units in a layer, since any rotation of the units can change the correlation between units without affecting the objective function of SNI regression.

The situation is different for the standard dropout ($\Sigma = \mathbf{I}$) and ASNI with $\Sigma = C$, as the penalty $w^\top (C \odot \Sigma) w$ is not invariant by rotation anymore. Interestingly, the following holds:

Lemma 3. *For $\Sigma = \mathbf{I}$ or $\Sigma = C$, where C is the covariance matrix of a set of points $x_1, \ldots, x_n \in \mathbb{R}^d$, it holds that:*

1. $\forall i, j \in [1, d], (C \odot \Sigma)_{ij} \geq 0$,
2. $\sum_{i,j=1}^d (C \odot \Sigma)_{ij}$ *is invariant by rotation of the points.*

Proof. For the first point, just notice that for $\Sigma = \mathbf{I}$, the entries of $C \odot \Sigma$ are $C_{ii} \geq 0$ on the diagonal, and 0 elsewhere; for $\Sigma = C$, the entries of $C \odot \Sigma$ are $C_{ij}^2 \geq 0$. For the second point, note that for $\Sigma = \mathbf{I}$, the sum considered is just the trace of the covariance matrix C, which is invariant by rotation (sum of eigenvalues); for $\Sigma = C$, the sum considered it the squared Frobenius norm of C, which is also invariant by rotation (sum of the squares of eigenvalues). □

As the penalty induced by SNI is

$$\lambda w^\top (C \odot \Sigma) w = \lambda \sum_{i,j=1}^d (C \odot \Sigma)_{ij} w_i w_j \,, \tag{4}$$

Lemma 3 suggests an interplay between the optimization of the inputs (which impact $C \odot \Sigma$) and the optimization of w. If we fix w and just optimize over a rotation of the inputs, then the penalty (4) is just a linear function in $C \odot \Sigma$, which according to Lemma 3 stays in a linear polyhedron defined by linear equalities and inequalities, and one might expect the best rotation to push $C \odot \Sigma$ near the boundary of that polyhedron, where some entries are 0. Of course a more careful analysis is needed to make this reasoning rigorous (in particular, w should also be rotated, and $C \odot \Sigma$ can not span the whole polyhedron), but it may hints that both dropout and ASNI with $\Sigma = C$ tend to create representations with small values in $C \odot \Sigma$. While this only concerns variance terms for usual dropout, a possible benefit of ASNI with $\Sigma = C$ is that it involves all off-diagonal terms C_{ij}^2 as well, suggesting that ASNI may create less correlated representations by penalizing the correlation among units of a given layer. We study this effect more precisely in the experiments below.

6 Experiments

6.1 Simulation

In order to study the performance of ASNI on a toy model, we use the classical simulation setting proposed by Guyon et al. (2007) for the MADELON dataset. In short, we generate 100 samples for training and 10 000 test samples from 2 balanced classes, and train a linear model on the same training set using (1) no dropout, (2) i.i.d Gaussian dropout with different values of λ, and (3) ASNI using $\Sigma = C$ with different values of λ. The MADELON procedure allows to vary total number of features, as well as the number or redundant features. We report in Tables 1 and 2 the test accuracies of the different models, when we vary the total number of features on the one hand, and when we vary the number of redundant features on the other hand.

Table 1: Best average test classification score of a linear model without noise injection, with i.i.d Gaussian dropout, and with ASNI on the MADELON simulation with 10% useful features and no redundant features: varying the total number of features.

Redundant	**No drop.**	**i.i.d Gauss.**	**ASNI**
10^2	66.6 ±2.6	**68.3** ±2.0	68.0 ±2.0
10^3	68.1 ±2.3	68.6 ±2.0	**68.9** ±2.2
10^4	55.6 ±2.5	54.9 ±2.8	**56.2** ±2.6

Table 2: Best average test classification score of a linear model without noise injection, with i.i.d Gaussian dropout, and Structured dropout (ASNI) on the MADELON simulation with 1000 features and 100 useful : varying the number of redundant features .

Features	**No drop.**	**i.i.d Gauss.**	**ASNI**
0	66.6 ±2.3	**68.3** ±2,0	68.0 ±2.0
100	65.8 ±1.6	67.6 ±1.3	**68.2** ±1.3
800	68.9 ±1.6	69.1 ±1.6	**71.6** ±1.6

We notice that in most settings ASNI performs best, particularly when the total number of features grows. This suggests that ASNI acts as an effective regularizer even for linear models. It also significantly stands out in the presence of redundant features. An intuition is that ASNI allows us to use the weights of the redundant features in accordance to the useful features they are created from and minimises prediction disagreement among single weights (since it ties weights in the regularisation).

6.2 MNIST

We now assess the performance of ASNI on image classification, using the classical MNIST benchmark. For simplicity, we train a network with only 2 dense ReLU-activations hidden layers. We do not expect to obtain state-of-the-art results as we do not perform any data augmentation or other regularisation. The goal of this set of experiments is mainly to study the difference of performance and the effect of ASNI on a hidden layer activations compared with independent noise injection. The number of the second hidden layer units $d^{(2)}$ is fixed to the number of classes (10 here), and we vary the number of units in the first hidden layer $d^{(1)}$.

We summarize in Table 3 the test accuracy defined as the proportion of well classified examples from the test set, after training the 2 hidden layer network with varying number of units. Figure 2 shows the evolution of this test error during the training process of the model with 256 units in the hidden layer, as a function of the number of SGD iterations.

We see from Table 3 that all methods involving noise injection tend to outperform the baseline approach without no regularization, which confirms the benefits of noise injection for performance. Second, we notice

Table 3: Best average classification score on the MNIST dataset of a 2 hidden layer without noise injection, with i.i.d noise injection (Gaussian and Bernoulli dropout), and with ASNI, when we vary the number of units in the first hidden layer.

$d^{(1)}$	**No drop.**	**iid Gauss.**	**iid Bern.**	**ASNI**
32	93.7 ± 0.2	94.2 ± 0.8	94.4 ± 0.9	**95.8** ± 0.4
64	95.8 ± 0.6	95.4 ± 0.7	95.9 ± 0.6	**96.6** ± 0.7
256	96.1 ± 0.6	97.0 ± 0.7	97.4 ± 0.7	**97.8** ± 0.7
512	96.5 ± 0.1	97.5 ± 0.1	97.6 ± 0.1	**98.1** ± 0.1
1,024	96.2 ± 0.1	97.6 ± 0.1	97.6 ± 0.2	**98.1** ± 0.3

that among the three methods that perform noise injection, ASNI constantly outperforms both Gaussian and Bernoulli i.i.d dropout for small and large number of units. The 2 hidden layers, that seems to overfit even for a small number of units in both hidden layers, has however a better accuracy with larger number of units, which indicates that there is still information to be gained from the data. The network with 64 units however, with ASNI regularization, seems to capture more information than the network without dropout with 1024 units, which can be largely explained by the quality of representation learnt by structured dropout, as we will show below. Figure 2 also shows that ASNI leads to faster convergence, an effect observed as well in batch normalisation (Ioffe and Szegedy, 2015).

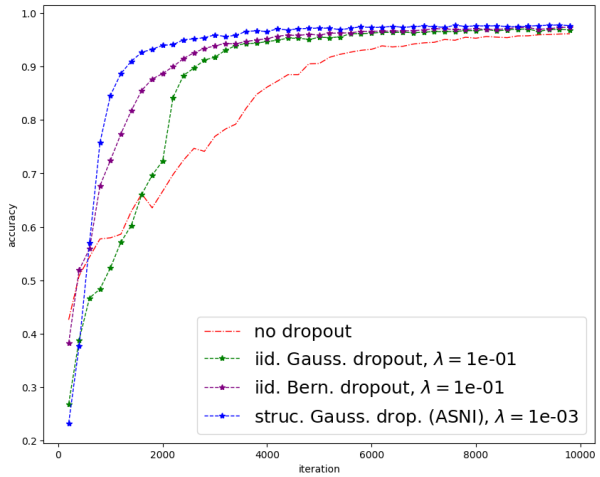

Figure 1: Test classification during training for a 2-hidden layers MLP trained without noise injection, with i.i.d. noise injection or with ASNI (with the best regularisation hyper-parameter), with 256 units in the hidden layer.

To see the effects of ASNI on units co-adaptation we measure the total correlation of the units' activations at a layer l as the Frobenius norm of the activations correlations matrix T defined as:

$$T_{ij} \frac{\Sigma_{i,j}}{\sqrt{\Sigma_{i,i}\Sigma_{j,j}}},$$

where Σ is the covariance of the unit activations introduced in methods description. The evolution of this quantity is shown in Figure 2, for the network with 1,024 units in the hidden layer. We see that for all methods, correlations among units tends to decrease during optimization, which confirms that better performance is obtained when units are less redundant. We also see that adding noise has a dramatic effect on the decrease

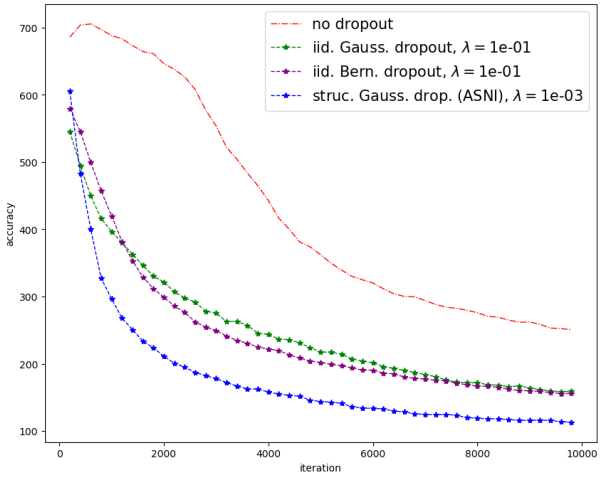

Figure 2: Correlation matrix norm of the first hidden layer activations during training for a 2-hidden layers MLP trained without noise injection, with i.i.d. noise injection or with ASNI (with the best regularisation hyper-parameter), with 1,024 units in the first hidden layer.

of correlation, as all three methods regularized by noise injection see their units' correlations decrease much faster and much lower than the unregularized baseline. Gaussian and Bernoulli i.i.d. noise injection lead to a very similar curve, confirming that both methods behave very similarly. Finally, we observe that ASNI lead s to faster decrease of the correlation matrix norm, and that it reaches after 10^4 iterations a lower value than all other methods. This empirically confirms the active role played by non-i.i.d. noise injection, in particular ASNI, in promoting non-redundant representations.

To further study the quality of the representations learned by different methods, we visualize the vectors of hidden layer activations on the test set using t-SNE in order to assess how well the different classes are separated. Figure 3 (right panels) shows the 2-component t-SNE embeddings of the second hidden layer activations (32 in this case) applied on a sample of 1,000 test samples, trained respectively without noise injection, with i.i.d. Gaussian or Bernoulli dropout, and with ASNI. Visually, we see that the class are better separated in the representation learned by ASNI than by the other methods. To quantify this visual impression, we measure the quality of the representation by computing the Silhouette coefficient of each t-SNE embedding (Rousseeuw, 1987). A larger Silhouette value indicates that the representation is better at recovering the known classes of images (Chen et al., 2002). We report in Table 4 the mean silhouette coefficients over all test samples for 10 clusters with respectively 32, 256 and 1,024 units in the first hidden layer. These results confirm what we qualitatively observed in Figure **??**, namely, that noise injection improves the quality of the representations compared to the non-regularized version, and more importantly that ASNI clearly outperforms i.i.d. noise injection in all settings.

Table 4: Average Silhouette coefficient scores of the last hidden layer t-SNE embeddings on MNIST test dataset, without noise injection, with i.i.d Gaussian and Bernoulli dropout, and structured dropout (ASNI).
.

$d^{(1)}$	**No drop.**	**iid Gauss.**	**iid Bern.**	**ASNI**
32	0.60	0.57	0.53	**0.69**
256	0.58	0.63	0.63	**0.72**
1024	0.58	0.73	0.74	**0.80**

In order to further investigate which layers are decorrelated by ASNI, we train the same 2 hidden layers

architecture on MNIST, but we apply independent noise injection or ASNI only on a single layer for each experiment (input, first hidden or second hidden layer). The evolution of the first and second hidden layer's correlations during training (represented again by the Frobenius norm of the activations correlation matrix at iteration t) for each experiment is shown in Figure 4. We see that i.i.d. Bernoulli and Gaussian dropout do not necessarily reduce the correlations between units, and thus do not always prevent co-adaptations in terms of activations correlations. ASNI, on the other hand, forces units to be more independent when it is applied on that layer, but does not reduce cross-correlations completely to 0 since the norm of the correlation matrix continues to decrease during the training. In this sense, ASNI is different from the whitening techniques mentioned in the introduction in that it does not explicitly change the input and does not force units to be independent such as batch normalisation and its decorrelated variants, but rather encourages units through the structure of dropout to be more independent. Interestingly, Figure 4 also shows that in the case of dense multilayer networks, applying ASNI on one layer does decorrelate the activations of that layer but not of the next layers. However, applying ASNI on the second hidden layer decorrelates both its activations and the activations of the first hidden layer.

Finally, we assess whether ASNI regularization has an effect on the sparsity of activations. Figure 5 shows the histogram of the activation values after training the same 2 hidden layers network without dropout, with i.i.d. gaussian or Bernoulli dropout, or with ASNI. It confirms the findings of Srivastava et al. (2014) that dropout may lead to sparser representations. However, we can see that ASNI provides a sparser activations distribution than dropout, while improving on accuracy as previously shown in Table 3. We also notice that Bernoulli dropout and its gaussian variant result in a similar level of sparsity, in this experiment at least, which leads to think that this effect is independent from the sparsity of the multiplicative noise itself.

6.3 CIFAR10 and CIFAR100

Finally, we compare ASNI to i.i.d. noise injection on a more realistic setting, namely, a LeNet convolutional network architecture with 4 convolutional layers followed by 2 dense layers tested on the CIFAR10 and CIFAR100 datasets. We again compare the different noise injection schemes applied on the 2 dense hidden dense layers, without data augmentation or additional regularization.

Table 5 summarizes the test accuracy reached by the different training procedures. We again observe that all noise injection methods outperform the baseline, that Gaussian and Bernoulli i.i.d. dropout behave very similarly, and that ASNI has the best performance for these datasets. We also notice that ASNI has less variance in performance compared to all other methods, which might be explained by the faster convergence observed already in MNIST experiments.

Table 5: Best average test classification score of LeNet on CIFAR10 without noise injection, with i.i.d. noise injection (Gaussian and Bernoulli dropout), and with ASNI on CIFAR10 and CIFAR100 benchmarks. .

Data	**No drop.**	**iid Gauss.**	**iid Bern.**	**ASNI**
CIFAR10	66.5 ± 0.1	67.9 ± 0.3	67.7 ± 0.4	**68.3 ± 0.2**
CIFAR100	32.9 ± 0.2	33.8 ± 0.5	33.8 ± 0.5	**34.4 ± 0.3**

As for the MNIST experiments, we also measure the amount of correlations between unit activations, evaluated by the Frobenius norm of the correlation matrix, and show how it evolves over training for the different methods in Figure 6. We notice that standard Bernoulli dropout has a weaker effect on reducing correlations than other methods on CIFAR10, but that overall all methods significantly reduce correlation during training. After convergence, ASNI keeps a small advantage on both datasets in terms of correlation level reached. As shown in Table 6, the representation learned by ASNI has also a larger Silhouette than other methods on the test sets.

Table 6: Average Silhouette coefficient of LeNet's last hidden layer t-SNE embeddings on CIFAR test datasets, without noise injection, with i.i.d Gaussian and Bernoulli dropout, and with ASNI. .

Data	**No drop.**	**iid Gauss.**	**iid Bern.**	**ASNI**
CIFAR10	0.38	0.43	0.42	**0.48**
CIFAR100	0.35	0.37	0.36	**0.38**

7 Conclusion

We proposed new regularization schemes that generalize dropout, by creating correlations between the noise components. We focused particularly on ASNI, an adaptive approach that replicates the structure of the data correlation in the noise correlation. We showed both theoretical and empirical results suggesting that ASNI improves the representation and performance of shallow and deep neural network, while maintaining computation efficiency. The ASNI framework opens new research directions. First, one way consider different ways to create the noise correlation structure, using for example the structure of the network, or may even think about *learning* it. Second, while Gaussien noise is convenient to impose a particular correlation structure, discrete noises such as binary noise can be computationally advantageous; sampling binary random variables with a given covariance matrix is however not an easy task (Leisch et al., 1998; Preisser and Qaqish, 2014), and progress in that direction may be directly useful for DNN regularization.

8 Availability

All code concerning the real data experiments is available at `https://github.com/BeyremKh/ASNI`

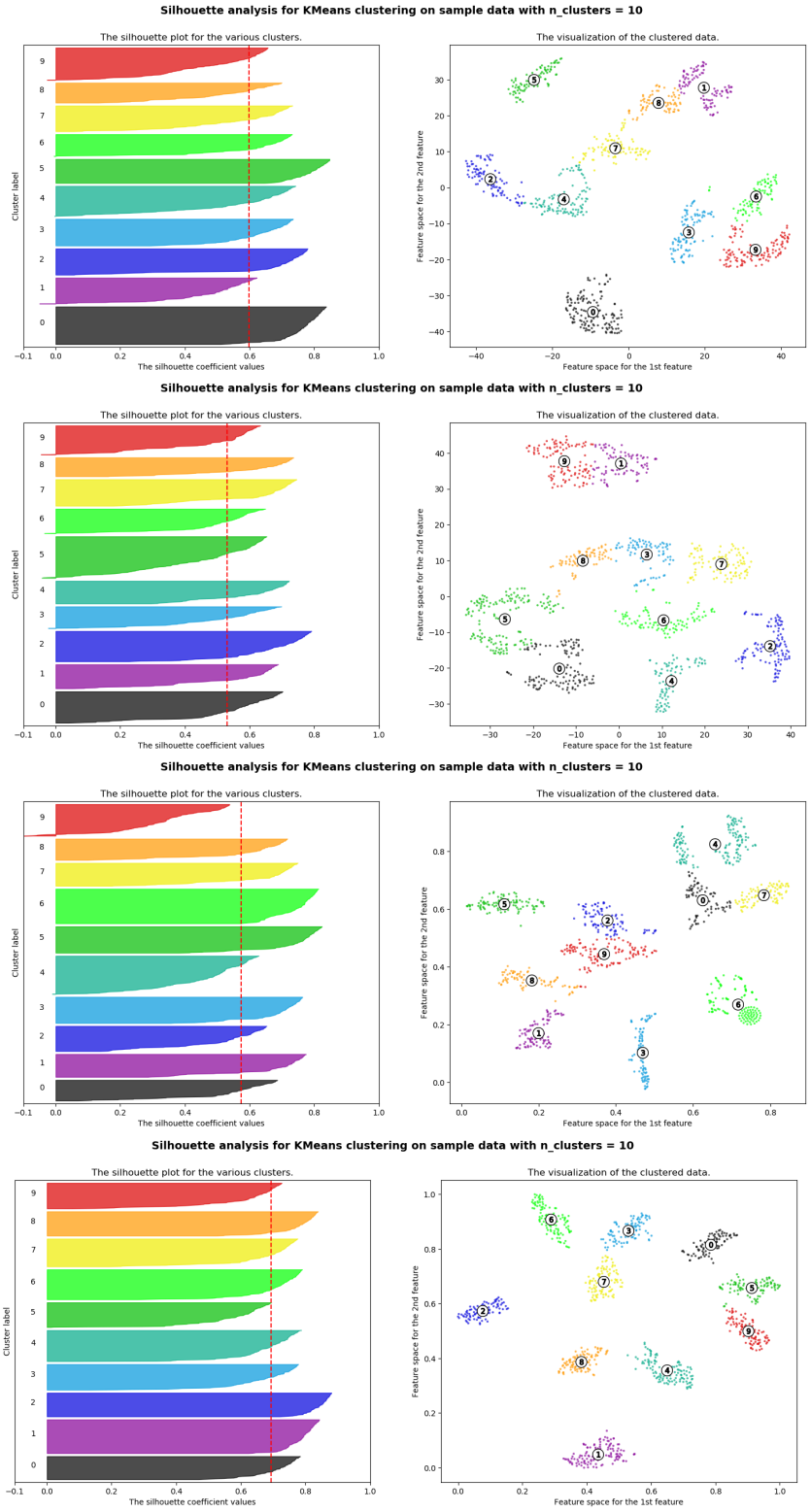

Figure 3: Silhouette plots (left) for the t-SNE embeddings (right) of the first hidden layer activations on 1,000 MNIST test images (2 hidden layers MLP with 32 units on the first layer). We compare, from the top to the bottom row, a network trained without noise injection, with i.i.d. Gaussian dropout, with i.i.d Bernoulli dropout, and with ASNI. The points are colored and numbered according to the class of the images.

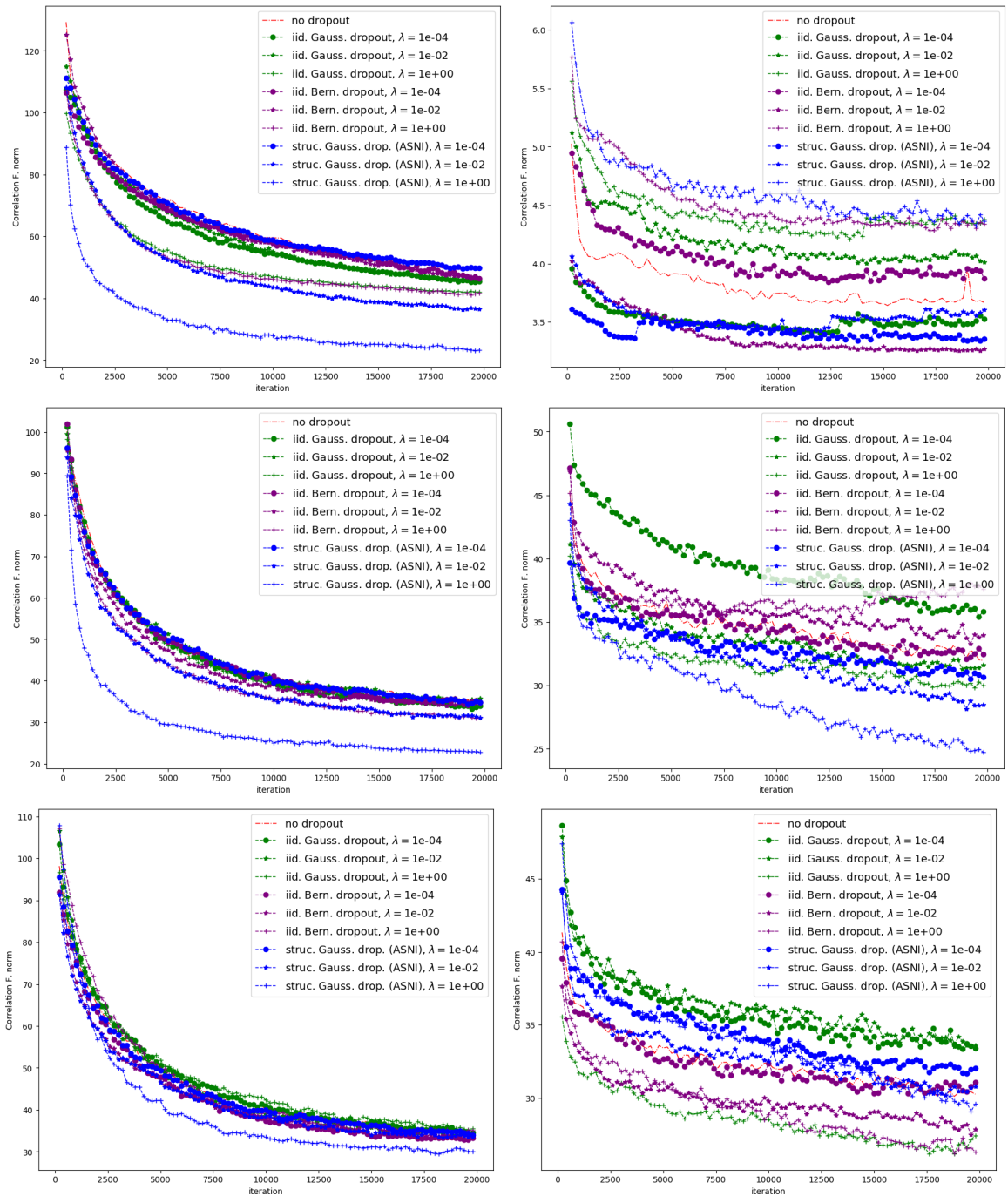

Figure 4: Correlation matrix norm of the first (left figures) and second (right figures) hidden layer activations during training for a 2-hidden layers MLP with no noise injection, with iid noise injection and ASNI, applied on the first hidden layer only (above), on the the second hidden layer only (middle) or on the input layer only (below).

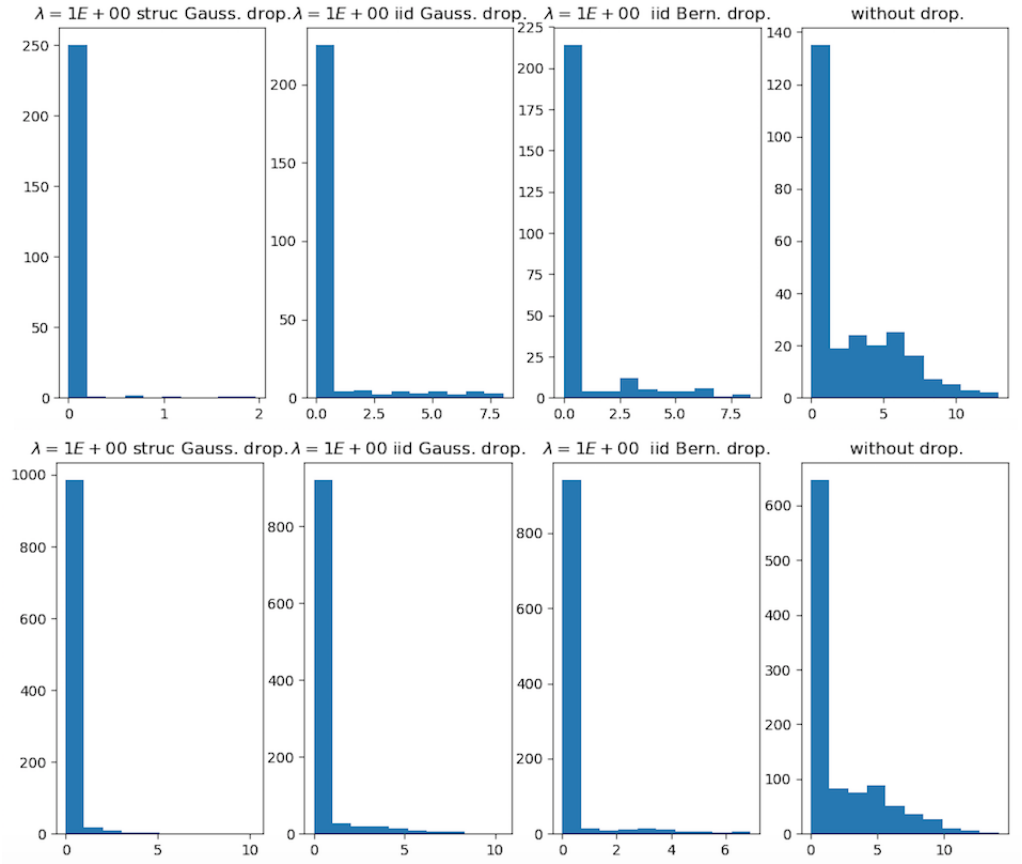

Figure 5: First layer activations after training our 2 hidden layers network on MNIST, without dropout, with i.i.d. gaussian dropout, i.i.d. Bernoulli dropout or structured dropout (ASNI). With 256 units (above) and 1024 units (below).

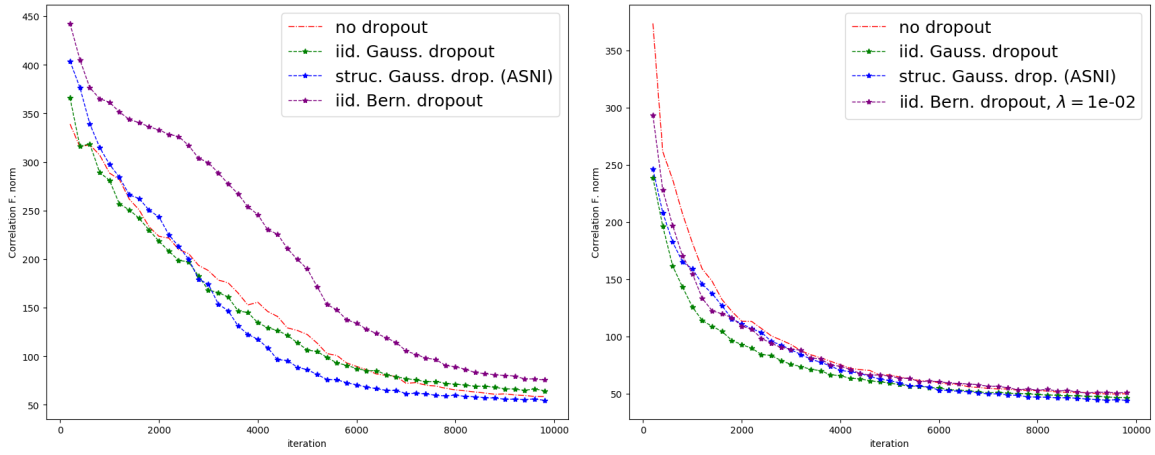

Figure 6: Correlation matrix norm of the first dense hidden layer activations with LeNet, with either no noise injection, iid noise injection or ASNI, during training on CIFAR10 (left) and CIFAR100 (right).

References

Sergul Aydore, Bertrand Thirion, Olivier Grisel, and Gael Varoquaux. Using feature grouping as a stochastic regularizer for high-dimensional noisy data. Technical Report 1807.11718, arXiv, 2018. URL `http://arxiv.org/abs/1807.11718`.

Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Adv. Neural. Inform. Process Syst.*, volume 26, pages 3084–3092. Curran Associates, Inc., 2013.

Pierre Baldi and Peter J Sadowski. Understanding dropout. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Adv. Neural. Inform. Process Syst.*, volume 26, pages 2814–2822. Curran Associates, Inc., 2013.

B BARLOW. Possible principles underlying the transformations of sensory messages. *Sensory Communication, Contributions: Contributions*, 217, 1959.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: a review and new perspectives. Technical Report 8, 2013. URL `http://dx.doi.org/10.1109/TPAMI.2013.50`.

Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1): 108–116, 1995.

Gengxin Chen, Saied A Jaradat, Nila Banerjee, Tetsuya S Tanaka, Minoru SH Ko, and Michael Q Zhang. Evaluation and comparison of clustering algorithms in analyzing es cell gene expression data. *Statistica Sinica*, pages 241–262, 2002.

Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. Technical report, 2015.

Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al. Natural neural networks. In *Advances in Neural Information Processing Systems*, pages 2071–2079, 2015.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. Technical report, 2017.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.

James E Gentle. *Computational statistics*, volume 308. Springer, 2009.

Isabelle Guyon, Jiwen Li, Theodor Mader, Patrick A Pletscher, Georg Schneider, and Markus Uhr. Competitive baseline methods set new standards for the nips 2003 feature selection benchmark. *Pattern recognition letters*, 28(12):1438–1444, 2007.

Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993.

David P Helmbold and Philip M Long. Surprising properties of dropout in deep networks. *The Journal of Machine Learning Research*, 18(1):7284–7311, 2017.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. Technical report, July 2012. URL `http://arxiv.org/abs/1207.0580`. arXiv: 1207.0580.

Aapo Hyvärinen. Independent component analysis: recent advances. *Phil. Trans. R. Soc. A*, 371(1984): 20110534, February 2013. ISSN 1364-503X, 1471-2962. doi: 10.1098/rsta.2011.0534. URL `http://rsta.royalsocietypublishing.org/content/371/1984/20110534`.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015. URL `http://dl.acm.org/citation.cfm?id=3045118.3045167`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017. ISSN 00010782. doi: 10.1145/3065386. URL `http://dl.acm.org/citation.cfm?doid=3098997.3065386`.

Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

Friedrich Leisch, Andreas Weingessel, and Kurt Hornik. On the generation of correlated artificial binary data. 1998.

Ping Luo. Learning deep architectures via generalized whitened neural networks. In *International Conference on Machine Learning*, pages 2238–2246, 2017.

Shin-ichi Maeda. A Bayesian encourages dropout. Technical report, December 2014. URL `http://arxiv.org/abs/1412.7003`. arXiv: 1412.7003.

S. Mallat. Group invariant scattering. *Comm. Pure Appl. Math.*, 65(10):1331–1398, 2012. doi: 10.1002/cpa.21413. URL `http://dx.doi.org/10.1002/cpa.21413`.

Zelda Mariet and Suvrit Sra. Diversity networks. 2016.

Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226–1238, 2005.

John S Preisser and Bahjat F Qaqish. A comparison of methods for simulating correlated binary variables with specified marginal means and correlations. *Journal of Statistical Computation and Simulation*, 84(11):2441–2452, 2014.

Pau Rodríguez, Jordi Gonzalez, Guillem Cucurull, Josep M Gonfaus, and Xavier Roca. Regularizing cnns with locally constrained decorrelations. Technical report, 2016.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Naftali Tishby and Noga Zaslavsky. Deep Learning and the Information Bottleneck Principle. Technical report, March 2015. URL `http://arxiv.org/abs/1503.02406`. arXiv: 1503.02406.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christopher Bregler. Efficient Object Localization Using Convolutional Networks. Technical report, November 2014. URL `http://arxiv.org/abs/1411.4280`. arXiv: 1411.4280.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Adv. Neural. Inform. Process Syst.*, volume 26, pages 351–359. Curran Associates, Inc., 2013.