Zero-bias, Denoising, Contractive Auto-Encoder On CIFAR Database

Badreddine Ben Nouma, Mouna Mokaddem
Department of Computer Science and Operations Research
University of Montreal
Montreal, Canada/Quebec, H3C 3J7
badreddine.ben.nouma@emt.inrs.ca
mouna.mokaddem@umontreal.ca

Abstract

The idea of our project is to use zero-bias[1], denoising[2], contractive[3] and a simple Auto-encoder [4] on CIFAR dataset for extracting features and pre-training both a feedforward neural network and a logistic regression in order to evaluate the classification task. We will compare performances of both classifiers as well as the manner used to build models (i.e., the structure of the implementation). Our focus will be on exploring single-hidden layer AE which have only one hidden layer as opposed to deep AEs which have multiple hidden layers.

1 Introduction

Classification of images turns out to be a hard task as it is not easy to find labeled images. However, un-labeled data (i.e., images) seem to be quiet available. Here comes the role of an AE which represents a neural network performing unsupervised learning. In fact, an AE captures the main factors of variation in the data and thus applying it on images, it will extract hidden features of that un-labeled data. Hence, AE are able to automatically extract meaningful features from data and to leverage the availability of un-labeled data to improve performance for tasks for which we have a little amount of label data[Bengio(2009)].

2 Definitions and reported previous work

2.1 Auto-encoder (AE)

2.1.1 Definition

An AE is a feedforward neural network trained to reproduce its input at the output layer where the input layer and the output layer have the same size (i.e., the same number of neurons). He takes the input data and maps it to a hidden layer which respect to:

$$y = f_\theta(\mathbf{X}) = s(W\mathbf{X} + b) \qquad with : \theta = (W, b) \tag{1}$$

[1]ZbAE: Zeros bias Auto-Encoder
[2]DAE: Denoising Auto-Encoder
[3]CAE: Contractive Auto-Encoder
[4]AE: Auto-Encoder

where «s» is a nonlinear activation function typically a logistic sigmoid (2).

$$s(W\mathbf{X}+b) = \frac{1}{1+\exp(-(W\mathbf{X}+b))} \tag{2}$$

The resulting latent representation «y» is then mapped back to a "reconstructed" vector «$\hat{\mathbf{X}}$» which respect to (3):

$$\hat{\mathbf{X}} = g_{\theta'}(\mathbf{y}) = s(W^{'}\mathbf{y}+b^{'}) \qquad with: \theta^{'} = (W^{'} = W^{T}, b) \tag{3}$$

The network is trained such that to minimize the reconstruction error which respect to the loss function (4) i.e. reproduce as perfectly as possible at the output layer the value that was fed in the input. After training, we will compare the values produced by the output layer «$\hat{\mathbf{X}}$» of the AE with values of the input layer «$\mathbf{X}$». The AE is encouraged to reproduce as perfectly as possible at the output layer the value that was fed in the input. In our project, as we work with RGB images, we will be using the loss function with real-valued inputs given by (4) .

$$\mathbf{J_A} = \frac{1}{2}\sum_{i=1}^{N} ||\mathbf{X_i} - \hat{\mathbf{X}}_\mathbf{i}||^2 \tag{4}$$

The structure of AE is as indicated by Figure 5.

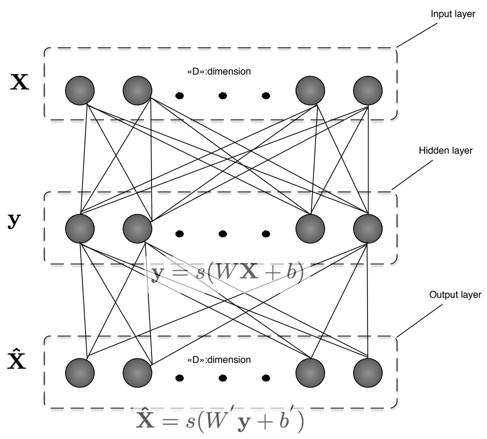


Figure 1: A single hidden layer AE

2.1.2 Loss function

The hope in an AE is to minimize a loss function [Bengio(2009)]. This can be done by comparing and to measure how good the reconstruction is. We train the AE on a training set by doing a gradient descent on that loss function given by (4). We distinguish two types of frequently used loss functions according to the type of the input as follows (i.e., binary and real-valued inputs)

2.1.3 The impact of the size of the hidden layer

The size of the vector of the encoder (i.e., the hidden layer) has an impact on the training of the AE in terms of the type of features or information that an encoder will learn about input distribution. In fact, we distinguish two cases where the first case is when the hidden layer is smaller than the input layer referred to as under-complete hidden layer whereas the second case is when the hidden layer is greater than the input layer, referred to as over-complete hidden layer. For an under-complete AE, we can notice that the hidden layer compresses the input as opposed to an over-complete AE whose encoder tend to map an input to its copy. However, experiments reported in [Bengio et al.(2007)Bengio, Lamblin, Popovici, and Larochelle] showed that, when trained with stochastic gradient descent, non-linear over-complete AE learn useful representation (i.e., a representation that gives low classification error when fed to a network as a classifier). To prevent an over-complete AE from learning the identity function, there are ways to force the encoder capture useful features about the input. Those ways define new variants of AE. We will discuss them in the next section.

2.2 Variants of AEs

2.2.1 Denoising AE

As we saw in the previous section, there is no guarantee that over-complete hidden layer's units would extract meaningful features of the input. To surround this problem, [Vincent et al.(2008)Vincent, Larochelle, Bengio, and Manzagol] constructed a new variant of AE referred to as DAE. The idea of the DAE is to learn a representation extracted by the encoder that is robust to introduction of noise. Thus, instead of feeding the encoder the original input , we will feed a noisy version of it as indicated in figure 2. The noisy version is obtained by taking the original input and passing it through a noise process that could be a probabilistic process which assigns a probability to the noisy version of the original input given the original input indicated by (5):

$$\tilde{\mathbf{X}} = q_D(\tilde{\mathbf{X}}|\mathbf{X}) \tag{5}$$

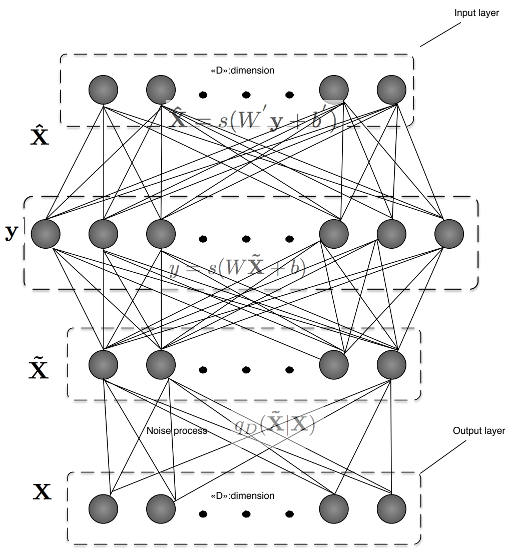


Figure 2: A single hidden layer AE

To train the DAE, we first compute the reconstruction from the corrupted input and then determine the loss function (6) which compares with the noiseless input.

$$\mathbf{J_D} = \frac{1}{2}\sum_{i=1}^{N} ||\tilde{\mathbf{X}}_\mathbf{i} - \hat{\mathbf{X}}_\mathbf{i}||^2 \tag{6}$$

Thus the hidden units of the DAE will be forced to extract particular types of correlation that we tend to see in the input vector of the training distribution. By doing this, the behaviour of just copying each input of the AE, with an over-complete hidden layer , will be broke as there is noise in it.

2.2.2 Contractive AE

As DAEs, contractive AEs come also to get round the problem of over-complete hidden layer's units. In fact, contractive AEs come up with an alternative approach to avoid uninteresting solutions by adding an explicit term in the loss function that penalizes those solutions. The main motivation of contractive AE is to extract features that only reflect variations observed in the training set. In other words, hidden units should be sensitive to variations that are specific to the training set and invariant to the other variations. The way to do this is to define a new AE loss function which contains, in addition to the error reconstruction of the AE, a new term that is defined as the square of the forbenius norm of the Jacobian of the encoder weighted by some parameter. The Contractive auto-encoder (CAE) is obtained with the regularization term of (7) yielding objective function.

$$\mathbf{J_C} = \frac{1}{2}\sum_{i=1}^{N} ||\mathbf{X_i} - \hat{\mathbf{X}}_\mathbf{i}||^2 + \lambda \sum_{i=1}^{N}\sum_{k=1}^{D} (\frac{\partial h_k(\mathbf{X})}{\partial \mathbf{X_i}})^2 \tag{7}$$

Note here that the error reconstruction is the term that enables the encoder to keep good information of the original input. Whereas the second term (i.e., the jacobian of the encoder) makes the encoder throw away all information. In fact, to minimize the square of the partial derivative, we have to have a partial derivative equal to zero and if the partial derivative is equal to zero, it means that if we change the value of the input, it will not change the value of the hidden unit and so it does not contain information about a particular element of the input. Thus, if we want the partial derivative to be zero for all the hidden units, we would have a hidden vector which does not vary if we change the input. Then, if we combine both terms, we will have an encoder that has hidden units that only keep useful information for the reconstruction of the original input.

2.2.3 Zero-bias AE

In 2.1.1 , an AE was defined as a model that is used for learning features from inputs where it is based on minimizing a loss function between an observation and a non-linear reconstruction «$\hat{\mathbf{X}}$» defined as:

$$\hat{\mathbf{X}} = F_v(W^{'} F_h(W\mathbf{X} + b) + b^{'}) \tag{8}$$

Where weight is a vector and is a bias for a certain hidden unit, is a vector of visible biases and is a hidden unit activation function. [Memisevic et al.(2014)Memisevic, Konda, and Krueger] made an empirical observation which assures that when training an AE with regularization, hidden unit biases take on large negative values. [Memisevic et al.(2014)Memisevic, Konda, and Krueger] recently showed that negative biases impede the learning of data distributions which have intrinsic dimensionality. In fact, during learning, by setting non-zero biases to large negative values as a result of regularization, an AE avoid trivial representations. However, these non-zero biases have a negative impact on the capacity of the model to learn a non-trivial density. For that reason, [Memisevic et al.(2014)Memisevic, Konda, and Krueger] suggested setting all biases to zero at the testing process «where there is no reason to constrain the model capacity in any way». Moreover, [Coates et al.(2011)Coates, Ng, and Lee] et [Saxe et al.(2011)Saxe, Koh, Chen, Bhand, Suresh, and Ng], « showed that very good classification performance can be achieved using a linear classifier applied to a bag of feature, using RELU activation without bias».

3 Experiments

3.1 Tested models

In our experiments we focused on evaluating four different variants of AEs applied on CIFAR dataset. For the task of evaluation, we first compare the filters images then use two different classifiers which are Feedforward neural network and a logistic regression.

3.2 Dataset used

We chose the CIFAR-10 dataset to study the ability of different variants of AEs to learn from high dimensional input data. CIFAR-10 dataset consists of color images in classes, with images per class. There are training images and test images [Krizhevsky and Hinton(2009)Krizhevsky, and Hinton].

3.3 Feed forward Neural network

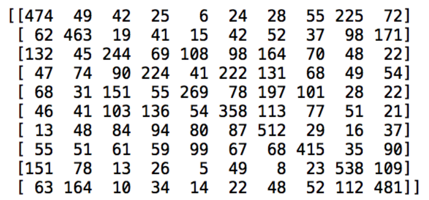
At the beginning the test is carried out on CIFAR without designing AE variants. For the task of evaluation we choose to do classification with a feedforward neural network.
In a first step, we applied PCA algorithm on learning data to reduce the dimensionality of the workspace, then we continue the data pre-processing with a «whitening stage» to remove some information from the processed signal. In a second step, we continue through the instantiation of the model (such as AE variants). Then we use data obtained after the pre-processing phase for learning model. Our classifier (i.e., a feedforward neural network) are initialized with the values of the learned weight matrix by the model. For the update process of network's parameters (w_1, b_1, w_2, b_2), we distinguished two cases. In the first case, we opted for a full update of all parameters which means updating and denoted by **full update case**. In the second case, we opted for a partial update which we update only «w_1» and denoted by **partial update case**.

In the first experiment, we are interested in the evaluation of our feed forward neural network on CIFAR. The result of this evaluation is showed in Table 2.

Table 1: Feed forward neural network on CIFAR

FEATURES	**DESCRIPTION**
Inputs&Outputs	1024*3
Hidden Layers	1 layers with 100 neurons
Learning-rate	0.001
Classification rate	39.78%

Another way to evaluate the performance of the network for each class of CIFAR dataset is the confusion matrix which allows visualization of the performance of an algorithm, typically a supervised learning one. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.



```
[[474  49  42  25   6  24  28  55 225  72]
 [ 62 463  19  41  15  42  52  37  98 171]
 [132  45 244  69 108  98 164  70  48  22]
 [ 47  74  90 224  41 222 131  68  49  54]
 [ 68  31 151  55 269  78 197 101  28  22]
 [ 46  41 103 136  54 358 113  77  51  21]
 [ 13  48  84  94  80  87 512  29  16  37]
 [ 55  51  61  59  99  67  68 415  35  90]
 [151  78  13  26   5  49   8  23 538 109]
 [ 63 164  10  34  14  22  48  52 112 481]]
```

Figure 3: Confusion matrix of a simple feed forward neural network on CIFAR

In the table below 2, we present the best values of hyper-parameters found during the testing phase, and the evaluation results for different variants of AE.

In same task of evaluation, the reconstruction of the input image for all AE variants is shown in the following figure 4.

Table 2: Feed forward neural network results

	Parameters	**ZbAE**	**CAE**	**DAE**	**AE**
Partial update	**Features Number**	100	100	100	100
	Time Learning model	59.41m	71.21m	69.54m	60.03m
	Learning rate of model	0.01	0.05	0.05	0.01
	Contraction	None	1.0	None	None
	Corruption	None	None	0.5	None
	Classification Rate	41.15%	39.53%	39.7%	38.74%
Full update	**Features Number**	100	100	100	100
	Time Learning model	57m	69.11m	65.2	58.44m
	Learning rate of model	0.01	0.05	0.05	0.01
	Contraction	None	1.0	None	None
	Corruption	None	None	0.5	None
	Classification Rate	48.91%	49.14%	49.12%	48.63%

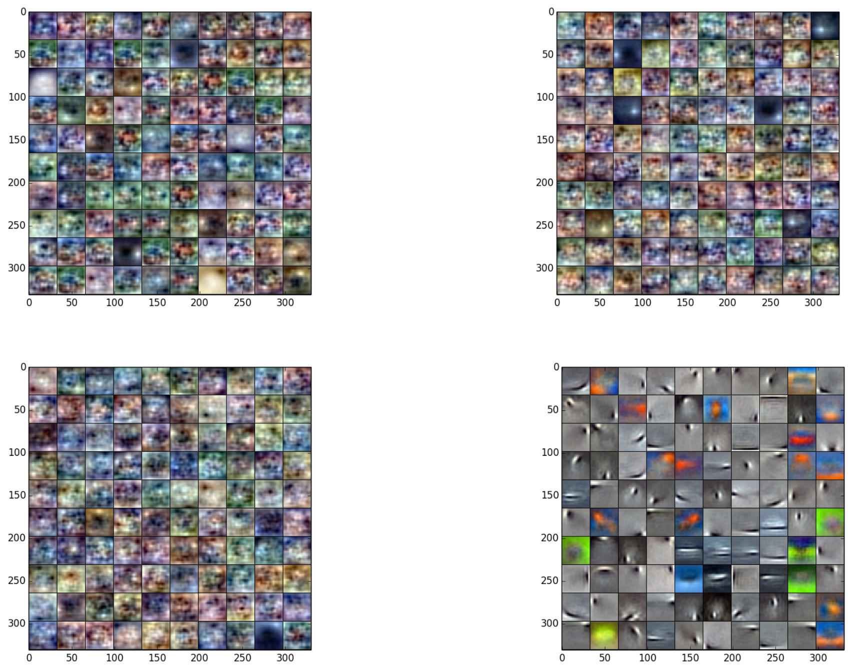


Figure 4: Top row: The right filter concerns the CAE model, the left filter concerns the AE model. Bottom row: The right filter concerns the ZbAE model, the left filter concerns the CAE model

We can conclude from the images of the filters obtained by ZbAE approach gives the best reconstruction of data than others. But in the context of classification. we can see that the success rate was not high enough, which brings us to think of settling over the hyper-parameter values in future work.

3.4 Logistic regression

The second implementation divides an image pixels into parts, each part is then divided into patches that may overlap. An AE is then fed with a patch of an image as an example. And after training all the patches, we reconstruct the image with the new representations of the patches (i.e., output of the AE) and give the latter to the classifier as shown by figure 3. The performance of an AE is then evaluated by a logistic regression. Note that all models were trained by stochastic gradient descent.

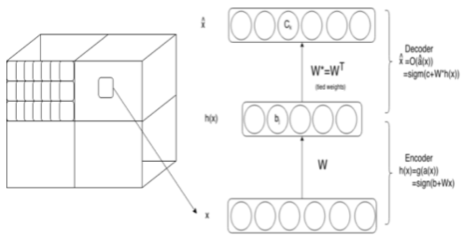


Figure 5: Training patches of images

The input data of size is contrast normalized and dimensionally reduced using PCA whitening. For all experiments of this section each model is trained for with a fixed momentum of (all parameters are indicated by table). For inference, we use sigmoid units. We classify the resulting representation using logistic regression with weight decay for classification. As a first approach to evaluate the different variants of AE is to look at the different filter images shown by 6. below which represent the reconstruction of the input image.

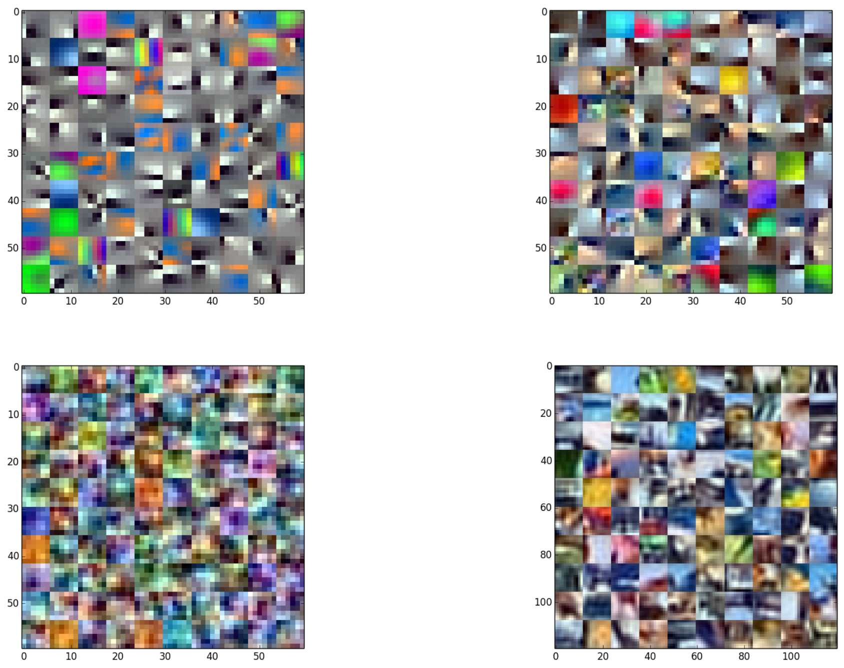


Figure 6: Top row: The left filter concerns the CAE model, the right filter concerns the DAE model. Bottom row: The left filter concerns the AE model, the left filter concerns the ZAE model

An interpretation of our observations could be that an AE that outputs a blurred image (i.e., filter) would give a higher error rate in the classification task since it failed to extract useful features of an input. However, observations cannot be reliable to determine which variant of an AE performs better. For that reason, we opted for using a logistic regression classifier which we applied on latent representations outputted by encoders. As indicated in table 1, we obtained best results with ZAE with test performance equal to 0.6254 versus 0.614 with CAE, 0.5719 with DAE and with 52.62 with AE.

Table 3: Logistic regression results

Parameters	**ZAE**	**CAE**	**DAE**	**AE**
Learning rate of model	0.01	0.05	0.01	0.01
Contraction	None	1.0	None	None
Corruption	None	None	0.5	None
Cost of the model	5.362589	5.446701	11.156237	0.757351
Classification cost	1.005763	1.050502	1.159788	1.294230
Classification rate	62.54%	61.4%	57.19%	52.62%

4 Discussion

Comparing results of the first implementation that uses a full image at a time as an input to an AE and results of the second implementation that uses every patch of an image as an input to an AE, we found that the second implementation gives higher classifying test performances. For instance for DAE, a classifying rate of 57.19 % versus 49.12%. The same for other variants of AEs. Although we did not do an exhaustive experimentation but our results show that the structure of dividing an image into patches and applying an AE on each patch of an image seems to extract a clearer latent representation of the input. And thus give higher classifying test performances. The initial intent of our project was to compare performances of different variants of AEs for extracting useful features from un-labeled data but after we find ourselves comparing different structures of implementations for the same variants. As a potential extension to our project, we can investigate deep AEs which have more than one hidden layer and compare their performances to single-layer AEs.

References

[Bengio(2009)] Bengio, Y. *Foundations and trends® in Machine Learning* **2009**, *2*, 1–127.

[Bengio et al.(2007)Bengio, Lamblin, Popovici, and Larochelle] Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H. *Advances in neural information processing systems* **2007**, *19*, 153.

[Vincent et al.(2008)Vincent, Larochelle, Bengio, and Manzagol] Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. Proceedings of the 25th international conference on Machine learning. 2008; pp 1096–1103.

[Memisevic et al.(2014)Memisevic, Konda, and Krueger] Memisevic, R.; Konda, K.; Krueger, D. *arXiv preprint arXiv:1402.3337* **2014**,

[Coates et al.(2011)Coates, Ng, and Lee] Coates, A.; Ng, A. Y.; Lee, H. An analysis of single-layer networks in unsupervised feature learning. International Conference on Artificial Intelligence and Statistics. 2011; pp 215–223.

[Saxe et al.(2011)Saxe, Koh, Chen, Bhand, Suresh, and Ng] Saxe, A.; Koh, P. W.; Chen, Z.; Bhand, M.; Suresh, B.; Ng, A. Y. On random weights and unsupervised feature learning. Proceedings of the 28th International Conference on Machine Learning (ICML-11). 2011; pp 1089–1096.

[Krizhevsky and Hinton(2009)Krizhevsky, and Hinton] Krizhevsky, A.; Hinton, G. *Computer Science Department, University of Toronto, Tech. Rep* **2009**,

- **Badreddine Ben Nouma**:
  1. 4 variants of AEs applied on a feedforward neural network
  2. Review the report
- **Mouna Moukaddem**:
  1. 4 variants of AEs applied on a logistic regression
  2. Redaction of the report