

# Introduction to Semi-Supervised Learning with Ladder Networks

JANUARY 19, 2016

Today, deep learning is mostly about pure supervised learning. A major drawback of supervised learning is that it requires a lot of labeled data and It is quite expensive to collect them. So, deep learning in the future is expected to unsupervised, more human-like.

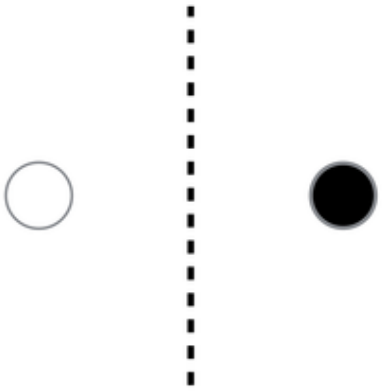
> "We expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object."
> – LeCun, Bengio, Hinton, Nature 2015

## Semi-Supervised Learning

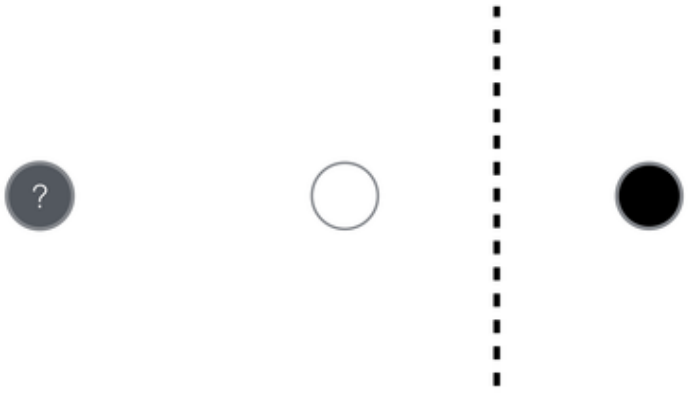
Semi-supervised learning is a class of supervised learning tasks and techniques that also make use of unlabeled data for training

- typically a small amount of labeled data with a large amount of unlabeled data.

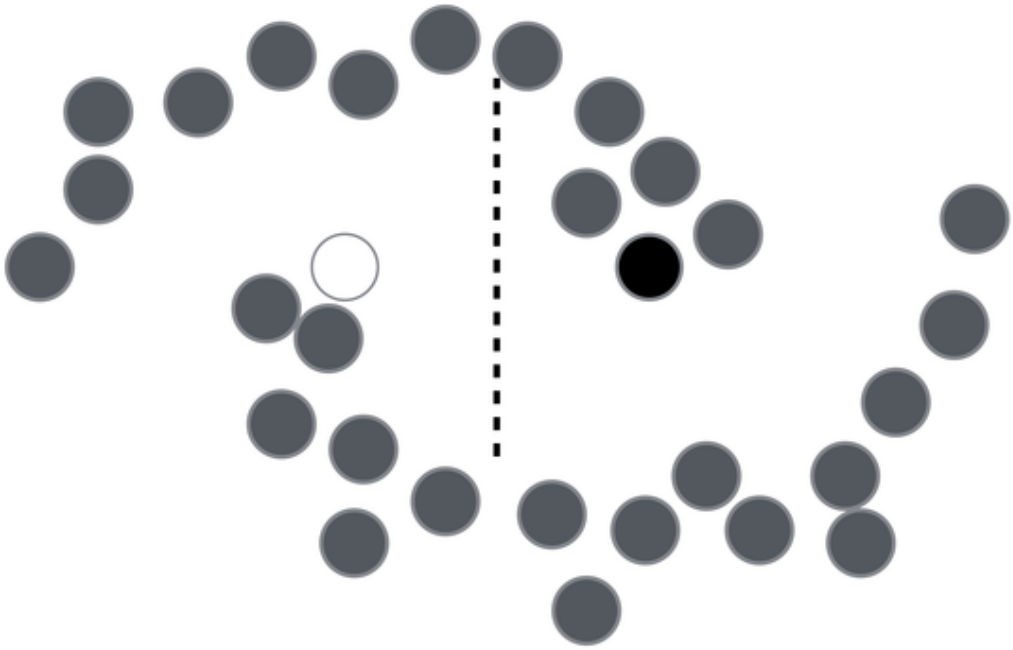
So, how can unlabeled data help in classification? Consider the following example (taken from these slides) consisting of only two data points with labels.



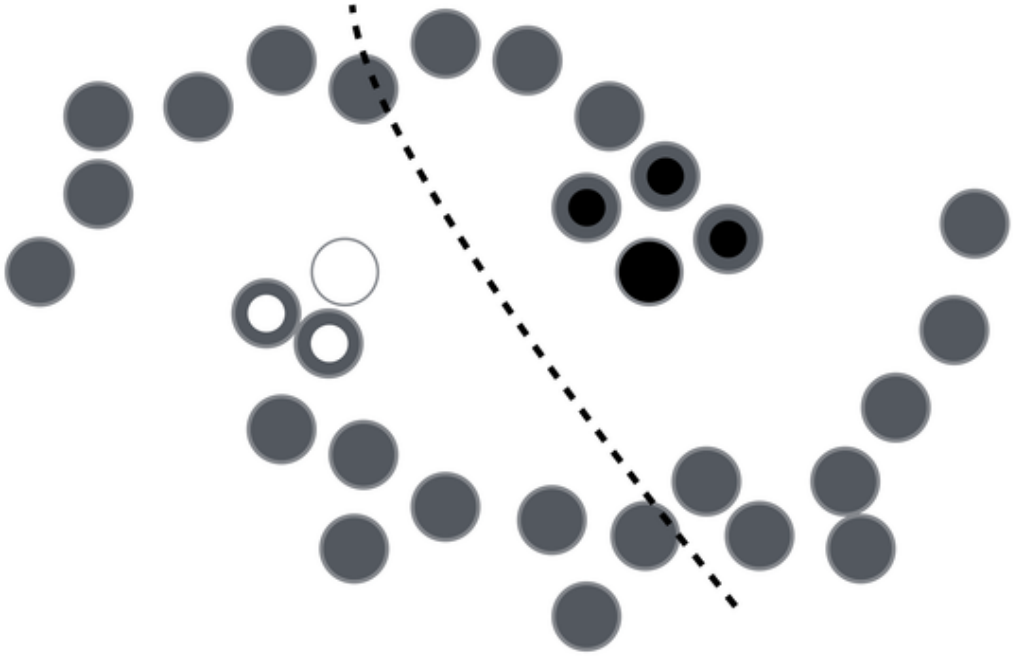
How would you label this point?

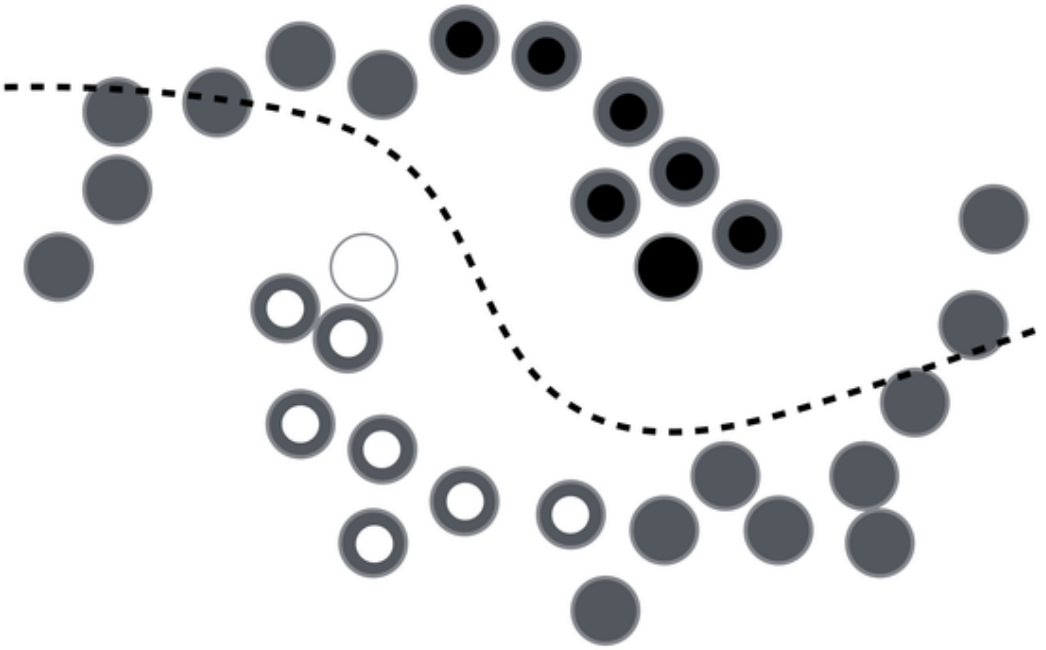


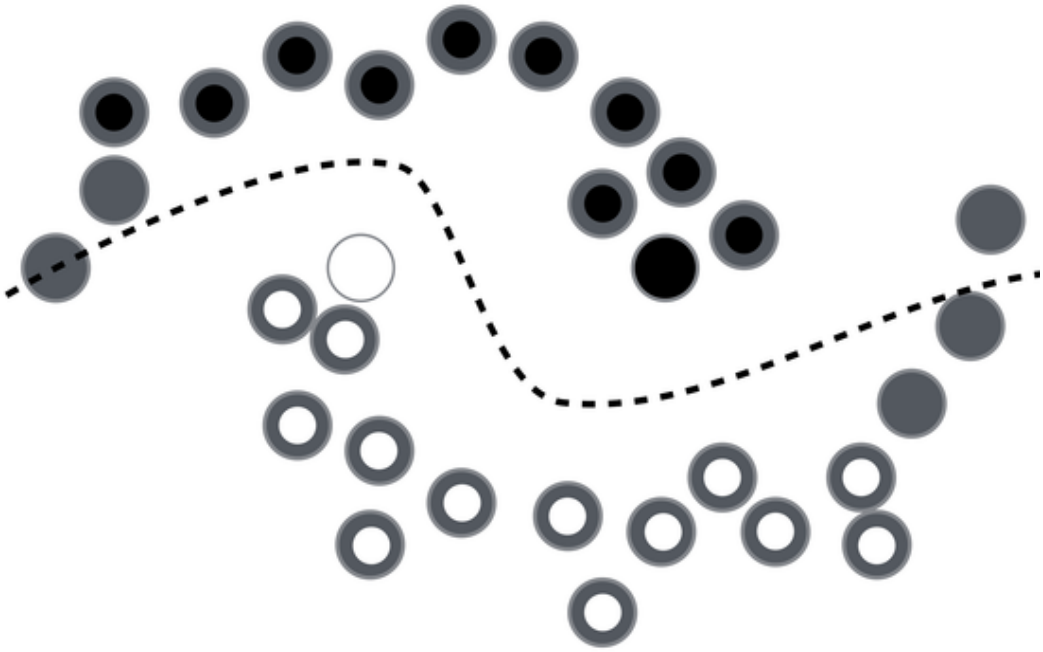

What if you see all the unlabeled data?

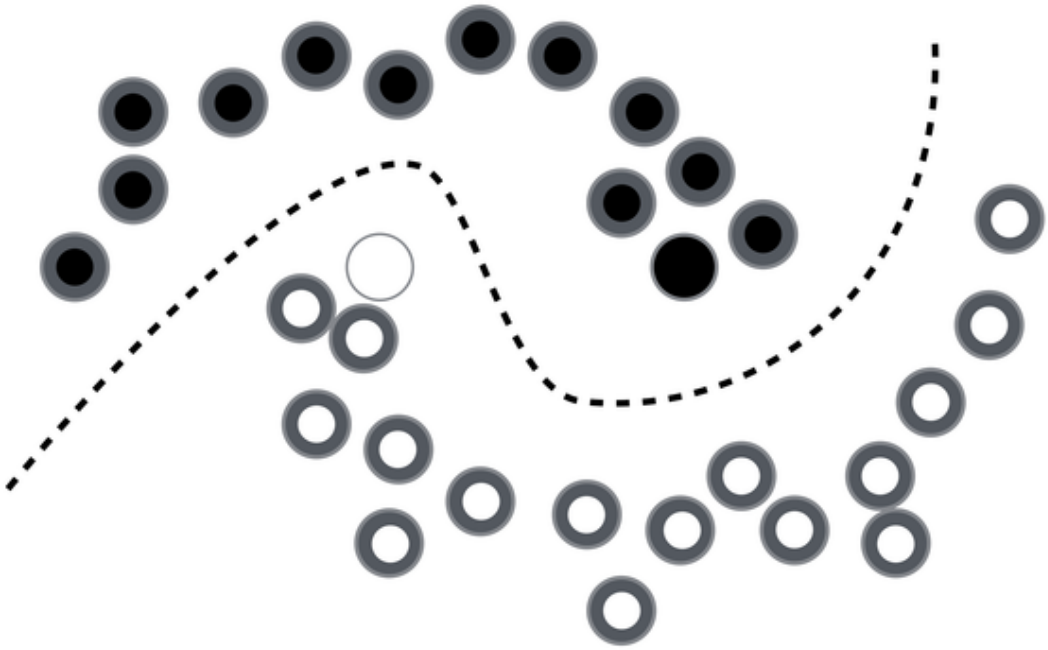


In order to make any use of unlabeled data, we must assume some structure to the underlying distribution of data. Labels are homogeneous in densely populated space ie., data points close to each other belong to the same class (smoothness assumption). Iterating this assumption over the data points until all data points are assigned a label,









It has been discovered that the use of unlabeled data together with a small amount of labeled data can improve accuracy considerably. The collection of unlabeled data is inexpensive relative to labeled data. Often labeled data is scarce and unlabeled data is plentiful. In such situations, semi-supervised learning can be used. Also, much of human learning involves a small amount of direct instruction (labeled data) combined with large amounts of observation (unlabeled data). Hence, semi-supervised learning is a plausible model for human learning.

## Ladder Networks

Ladder networks combine supervised learning with unsupervised learning in deep neural networks. Often, unsupervised learning was used only for pre-training the network, followed by normal supervised learning. In case of ladder networks, it is trained to simultaneously minimize the sum of supervised and unsupervised cost functions by backpropagation, avoiding the need for layer-wise pre-training. Ladder network is able to achieve state-of-the-art performance in semi-supervised MNIST and CIFAR-10 classification, in

addition to permutation-invariant MNIST classification with all labels.

## Key Aspects

#### Compatibility with supervised methods

It can be added to existing feedforward neural networks. The unsupervised part focuses on relevant details found by supervised learning. It can also be extended to be added to recurrent neural networks.

#### Scalability resulting from local learning

In addition to a supervised learning target on the top layer, the model has local unsupervised learning targets on every layer, making it suitable for very deep neural networks.

#### Computational efficiency

Adding a decoder (part of the ladder network) approximately triples the computation during training but not necessarily the training time since the same result can be achieved faster through the better utilization of the available information.

## Implementation

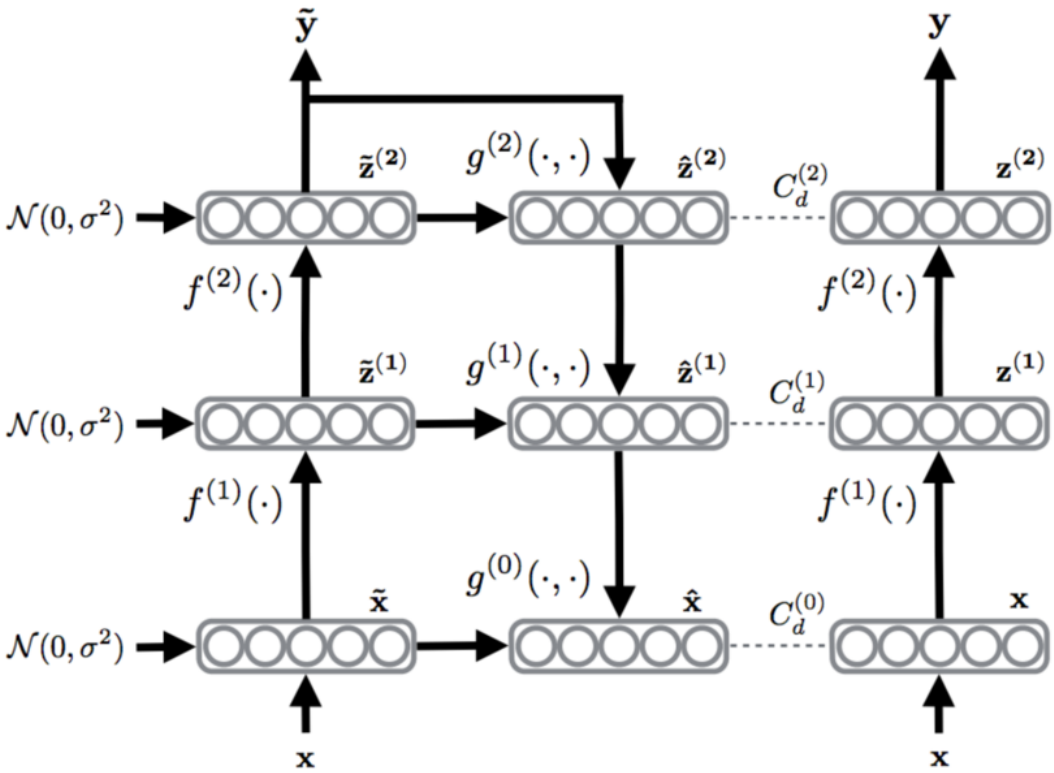
*This is a brief introduction to the implementation of Ladder networks. A detailed and in-depth explanation of Ladder network can be found in the paper "Semi-Supervised Learning with Ladder Networks".*

The steps involved in implementing the Ladder network are typically as follows:

1. Take a feedforward model which serves supervised learning as the encoder. The network consists of 2 encoder paths - clean and corrupted encoder. The only difference is that the corrupted encoder adds Gaussian noise at all layers.

2. Add a decoder which can invert the mappings on each layer of the encoder and supports unsupervised learning. Decoder uses a denoising function to reconstruct the activations of each layer given the corrupted version. The target at each layer is the clean version of the activation and the difference between the reconstruction and the clean version serves as the denoising cost of that layer.
3. The supervised cost is calculated from the output of the corrupted encoder and the output target. The unsupervised cost is the sum of denoising cost of all layers scaled by a hyperparameter that denotes the significance of each layer. The final cost is the sum of supervised and unsupervised cost.
4. Train the whole network in a fully-labeled or semi-supervised setting using standard optimization techniques (such as stochastic gradient descent) to minimize the cost.

An illustration of a 2 layer ladder network,




Batch normalization is applied to each preactivation including the topmost layer to improve convergence (due to reduced

covariate shift) and to prevent the denoising cost from encouraging the trivial solution (encoder outputs constant values as these are the easiest to denoise). Direct connection between a layer and it's decoded reconstruction are used. The network is called a Ladder network because the resulting encoder/decoder architecture resembles a ladder.

**Algorithm 1** Calculation of the output and cost function of the Ladder network

**Require:** $\mathbf{x}(n)$
# Corrupted encoder and classifier
$\tilde{\mathbf{h}}^{(0)} \leftarrow \tilde{\mathbf{z}}^{(0)} \leftarrow \mathbf{x}(n) + \texttt{noise}$
**for** l = 1 **to** L **do**
$\tilde{\mathbf{z}}^{(l)}_{\text{pre}} \leftarrow \mathbf{W}^{(l)}\tilde{\mathbf{h}}^{(l-1)}$
$\tilde{\boldsymbol{\mu}}^{(l)} \leftarrow \texttt{batchmean}(\tilde{\mathbf{z}}^{(l)}_{\text{pre}})$
$\tilde{\boldsymbol{\sigma}}^{(l)} \leftarrow \texttt{batchstd}(\tilde{\mathbf{z}}^{(l)}_{\text{pre}})$
$\tilde{\mathbf{z}}^{(l)} \leftarrow \texttt{batchnorm}(\tilde{\mathbf{z}}^{(l)}_{\text{pre}}) + \texttt{noise}$
$\tilde{\mathbf{h}}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\tilde{\mathbf{z}}^{(l)} + \boldsymbol{\beta}^{(l)}))$
**end for**
$P(\tilde{\mathbf{y}} \mid \mathbf{x}) \leftarrow \tilde{\mathbf{h}}^{(L)}$
# Clean encoder (for denoising targets)
$\mathbf{h}^{(0)} \leftarrow \mathbf{z}^{(0)} \leftarrow \mathbf{x}(n)$
**for** l = 1 **to** L **do**
$\mathbf{z}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)})$
$\mathbf{h}^{(l)} \leftarrow \texttt{activation}(\boldsymbol{\gamma}^{(l)} \odot (\mathbf{z}^{(l)} + \boldsymbol{\beta}^{(l)}))$
**end for**
# Final classification:
$P(\mathbf{y} \mid \mathbf{x}) \leftarrow \mathbf{h}^{(L)}$
# Decoder and denoising
**for** l = L **to** 0 **do**
**if** l = L **then**
$\mathbf{u}^{(L)} \leftarrow \texttt{batchnorm}(\tilde{\mathbf{h}}^{(L)})$
**else**
$\mathbf{u}^{(l)} \leftarrow \texttt{batchnorm}(\mathbf{V}^{(l)}\hat{\mathbf{z}}^{(l+1)})$
**end if**
$\forall i : \hat{z}^{(l)}_i \leftarrow g(\tilde{z}^{(l)}_i, u^{(l)}_i)$ # Eq. (1)
$\forall i : \hat{z}^{(l)}_{i,\text{BN}} \leftarrow \frac{\hat{z}^{(l)}_i - \tilde{\mu}^{(l)}_i}{\tilde{\sigma}^{(l)}_i}$
**end for**
# Cost function $C$ for training:
$\text{C} \leftarrow 0$
**if** $t(n)$ **then**
$\text{C} \leftarrow -\log P(\tilde{\mathbf{y}} = t(n) \mid \mathbf{x})$
**end if**
$\text{C} \leftarrow \text{C} + \sum_{l=0}^{L} \lambda_l \left\| \mathbf{z}^{(l)} - \hat{\mathbf{z}}^{(l)}_{\text{BN}} \right\|^2$

## Conclusion

The performance of Ladder networks is very impressive. On MNIST, it achieves an error rate of 1.06% with only 100 labeled examples! This is much better than previous published results, which shows that the method is capable of making good use of unsupervised learning. However, the same model also achieves state-of-the-art results and a significant improvement over the base-line model with full labels in permutation invariant MNIST classification, which suggests that the unsupervised task does not disturb supervised learning.

Ladder network is simple and easy to implement with many existing feedforward architectures, as the training is based on backpropagation from a simple cost function. It is quick to train and the convergence is fast, thanks to batch normalization.

## Code

The code published along with the original paper is available here - https://github.com/CuriousAI/ladder. My implementation of Ladder networks in TensorFlow is available here - https://github.com/rinuboney/ladder. *Note: The TensorFlow implementation achieves an error rate about 0.2% - 0.3% more than the error rates published in the paper. I am working on improving it.*

## Related papers

Semi-supervised learning using Ladder networks was introduced in this paper:

- Rasmus, Antti, et al. "Semi-Supervised Learning with Ladder Networks." Advances in Neural Information Processing Systems. 2015.

Ladder network was further analyzed and some improvements that attained an even better performance were suggested in this paper:

- Pezeshki, Mohammad, et al. "Deconstructing the Ladder Network Architecture." arXiv preprint arXiv:1511.06430 (2015).

and finally, these are the papers that led to the development of Ladder networks:

- Rasmus, Antti, Tapani Raiko, and Harri Valpola. "Denoising autoencoder with modulated lateral connections learns invariant representations of natural images." arXiv preprint arXiv:1412.7210 (2014).

- Rasmus, Antti, Harri Valpola, and Tapani Raiko. “Lateral Connections in Denoising Autoencoders Support Supervised Learning.” arXiv preprint arXiv:1504.08215 (2015).
- Valpola, Harri. “From neural PCA to deep unsupervised learning.” arXiv preprint arXiv:1411.7783 (2014).




**Mahmoud Su** • a year ago

nice job! I am just wondering whether Ladder Networks can be used in the absence of a fully labeled data set. Yes, it is trained on a limited number of labeled instances but evaluating the network takes a process using the entire supervised (labeled) data set. I am just curious since It could save a great time of hand-labeling new data sets.

• Reply • Share ›

**dissertation help uk** • 3 years ago

That type of concept on how could a person will going to learn that certain kind of subject looks helpful for them to be guided on what should be the necessary steps that they need to follow for them to learn effectively without using some excess time.

• Reply • Share ›

**Cristian Cardellino** • 3 years ago

Great post, and thank you for the implementation. I do have one question, in the code, as far as I can understand, and since you set the batch_size equal to the number of labeled examples, for every iteration step you take batch_size new examples of the unlabeled corpus, but the same batch_size examples from the labeled corpus with a permutation of these examples, am I right? You always need to do this? Or is a way you stop using the labeled examples at some point?

Thank you again for the post, is a really great work!

Thank you again for the post, is a really great work!

• Reply • Share ›

**rinuboney** Mod ➚ Cristian Cardellino • 3 years ago

Yes you are right. This is the way it is done in the official implementation. I haven't experimented much regarding this but I think you can get the same performance even if the batch size of the labeled examples is less than the current batch size or if they are only occasionally used. Intuitively I think the labeled examples are required to guide the network towards the particular supervised task. I don't remember seeing anything specific about this in any of the papers on ladder networks. I guess you can try it out for yourself.

Do let me know if you find anything cool :-)

1 • Reply • Share ›

**Matt Sevrens** • 3 years ago

It would be nice if the variable names in the tensorFlow code were a little less vague

• Reply • Share ›

**Lili Sun** • 3 years ago

Great post, I'd like to have a try:)

• Reply • Share ›

**Stephen** • 3 years ago

Great post - I think the link to Pezeshki et al. should be http://arxiv.org/abs/1511.0... (currently it goes to Rasmus et al. 2015)

• Reply • Share ›

**rinuboney** Mod ➚ Stephen • 3 years ago

Thanks. I've updated it.

• Reply • Share ›

Powered by Jekyll with Type Theme