Ukrainian Catholic University

Faculty of Applied Sciences

Data Science Master Programme

Unsupervised representation learning

Machine learning final project report

Authors:
Anton Shcherbyna
Vadym Korshunov

12 May 2019

Abstract

Deep learning plays an important role in development of intelligent systems nowadays. Application of deep neural networks is impossible without enormous amount of labeled data, hence unsupervised feature extraction from row data is very important direction in the research in recent years. In this work, we provide a detailed description of the representation learning, the importance and goals of this field, and discuss two state-of-the-art algorithms for learning representations - Deep InfoMax and Auto-Encoding Transformations. We report results of the experiments on the CIFAR10 and provide their discussion.

1 Introduction

1.1 Overview

Classic way of dealing with data modeling structured as follows: domain experts set the subjective "label" for certain data point as the "right" prediction for the model, and the we train the model to predict those labels based on data. Unfortunately, state-of-the-art deep learning models need a large amount of labeled data, and labeling is very expensive and long process. All this limits the applicability of deep learning in a lot of fields. Therefore, there has been an increasing interest in research to learn deep vector representations in an unsupervised way to solve data understanding tasks with small portion of labeled data.

A representation learning is the part of unsupervised and self-supervised learning, which is able to deal with unlabeled and unstructured data to obtain good results for particular task. Given large set of the high-dimensional data, we want to get the representation of this data in some low-dimensional space, so that it will capture some shared information or properties between different data samples. Also, we want those representations to be helpful for various tasks: classification, similarity search, etc. For example, texts and images, in general, very ambiguous, so finding the way to compress them into a vector with desired properties will simplify a lot of tasks with such types of data.

Let's look on the representation learning in terms of the following classification. In this case the main goal of the representation learning is to find better proxy - compressed vectors - for which optimization and prediction with labels will be better and quicker than constructing the entire model and training it from scratch. Also, the main consequence of applying successful representation are possibility of working with deep learning models almost or entirely without labels (finite number of classes or other type of response variable).

Nowadays there are a lot of ways to extract the representations, for example, get the output of the last layer from ResNet [1] or another popular neural network trained for classification. Or we can look at more sophisticated approaches: for example, we can use the generator and discriminator from GANs [2] or output of the last layer of the network trained to predict angle of the rotation, which manually made in [3].

In our work we'll describe and provide results of experiments for two recent state-of-the-art papers: Deep InfoMax and Auto-Encoding Transformations. We chose them, because they reported one of the best results to date and underline ideas behind those methods are extremely interesting.

2 Learning Deep Representations by Mutual Information Estimation and Maximization

At ICLR 2019 (International Conference on Learning Representations) will be presented a new algorithm - Deep InfoMax (DIM) [4], which uses the concept of mutual information to obtain meaningful represenations of tha data.

2.1 What is mutual information?

Mutual information is the concept, which lies on the intersection between information and probability theories. In simple words from information theory perspective, mutual information shows us how much information we can get about one variable, while observing another variable. And from probability theory perspective, mutual information measures how close joint probability between two variables is close to the product of their marginals. But let's look more formally. Let X and Y be two random variables, then:

$$I(X, Y) = D_{KL}(P_{xy}||P_x P_y),$$

where D_{KL} is a well-known in machine learning Kullback–Leibler divergence. Or more detailed:

$$I(X, Y) = \int_X \int_Y P_{xy}(x, y) \log \frac{P_{xy}(x,y)}{p_x(x) p_y(y)} dx dy$$

2.2 MI as training objective

So how we can use this concept to obtain useful representations? The answer is very simple, but in the same time - complicated. Basically, we can take some classic neural net like AlexNet or ResNet, cut the last fully-connected layer (like we do in transfer learning) and optimize the weights in such way, that they will maximize MI between last feature map and flattened features on the last layer. We'll call all the parts as the authors of the original paper do, so once again: encoder encodes an image into local feature map $C_\psi(x)$, then this feature map is summarized into global vector (representation that we need) via some transformation $E_\psi(x) = (C_\psi \circ f)(x)$. And finally we compute MI between local feature map and global vector, right? Not exactly. It turns out that we can't compute the MI directly, so that's why it is a bit complicated. But as always we can propose some tricky lower-bound and work with it, exactly this was proposed in paper [5]. I won't show all the derivations behind this approach, but to get some intuition let's look at this formula:

$$I(X, Y) = D_{KL}(P_{xy}|P_x P_y) \geq \hat{I}_\omega(X, Y) = \mathbb{E}_{P_{xy}}[T_\omega(x, y)] - \mathbb{E}_{P_x P_y}[e^{T_\omega(x,y)}],$$

where T_ω is some another neural net parametrized by ω. What we do here is basically introducing a lower-bound on the MI based on the the Donsker-Varadhan representation of the KL-divergence [6] with the help of a new network T_ω. So that now we can simultaneously optimize encoder and MI estimator:

$$(\hat{\omega}, \hat{\psi}) = \hat{I}_\omega(C_\psi(x), E_\psi(x)).$$

DIM introduces a couple of improvements [4] to this approach (e.g. share some parts of the network between encoder and MI estimator and use another divergence instead of KL), but you can find all the details in the original paper and I want to explain only the intuition behind this method.

But one crucial thing wasn't mention yet. Authors found out that the scheme described above (they called it global MI maximization) performs worse than slightly modified approach - local MI maximization. In the second option we optimize multiply MI between local patches $C_\psi(x)$ and global vector $E_\psi(x)$. Recall that $C_\psi(x)$ is a feature map obtained from one of the last layers of the network, so we can simply compute approximation of the mutual information between each patch $C_{\psi,i}(x)$ ($C_\psi(x) = \cup_i C_{\psi,i}(x)$) and $E_\psi(x)$, so the loss will be:

$$L(\psi, \omega) = \frac{1}{n^2} \sum_{i=1}^{n^2} \hat{I}_\omega(C_{\psi,i}(x), E_\psi(x))$$

This change was crucial for the improvement in the quality of the representations. The intuition behind this is pretty straightforward: as we force the global vector to have high MI with all local patches and as it has limited capacity, encoder would be forced to produce similar local patches (which will have high MI with global vector) and it's logical to assume that these similar patches will be determined by encoder based on some meaningful properties of the input data (e.g. spatial or class information).

2.3 Implementation details

For the encoder $E_\psi(x)$ I took ResNet18. This network is much smaller than original ResNet152, but still has enough capacity for CIFAR10 and also uses all the benefits of the residual connections, dropout and batch normalization. Network for MI estimation T_ω is a simple linear transformation.

3 Auto-Encoding Transformations

In 2019 Laboratory of Machine Perception and Learning (MAPLE) [7] presented state-of-the-art unsupervised learning algorithm [8]. Their research describes how update encoder-decoder scheme of data to encoder-decoder scheme of transformations of the data with adding specific loss for transformations.

3.1 Encoder-Decoder scheme

Empirically, representation learning is the process of obtaining the strong low-dimensional vector representations. With vectors, we can interchange the manipulation between data to manipulation between representations.

General framework to learn represenations is the decoder-encoder scheme:
Let we have $\mathbb{ENC}(.)$ - function that compress data into some other space with smaller dimension than original input, and $\mathbb{DEC}(.)$ - function, that transfer input argument to the space with equal dimensionality of input to the decoder. Let also we defined the distance between example and its restored variant as $l(x, \hat{x})$. So, the general of obtaining the representations procedure will be:

$$\sum_i l(x_i, \hat{x}_i(\theta)) \to \min,$$
$$\text{where } \hat{x}_i = \mathbb{DEC}(\mathbb{ENC}(x_i)),\ f = \mathbb{DEC} \circ \mathbb{ENC} \text{ has parameters } \theta$$

The procedure in some details can be differ, it is depends on domain and concrete task. But the main bottlenecks in scheme described above are constructing the proper compressing-decompressing

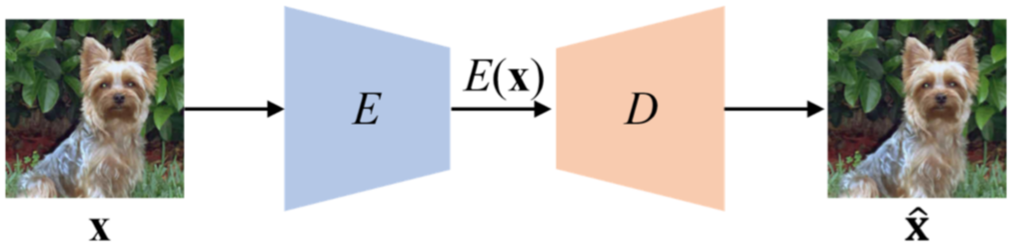

Figure 1: Encoder-Decoder scheme [8]

architecture, and constructing the loss between data samples. In case of images, we also can apply some transformations to the images, hence will expand the space of existed data. Hence, exist two general directions in improving encoder-decoder scheme:

- Loss constructing
- Parametrized feature extracting

3.2 The formulation

The main innovation of this paper - transformation of the data directly including in the architecture of encoder-decoder scheme. Authors constructed the parametrized family of transformations $\{\tau \in T\}$, which directly fitted into the model. The entire model will predict the parameters of transformation τ, and hence will boost the model to find the expressive representation of input image to approximate the transformation of the data. Intuitively, it will help, because, for example, we can define transformations as family of rotations, and in such case model must find features on the objects on the image that give additional information using the transformed data. Hence, model will learn features in unsupervised way, which can be used in future prediction tasks.

Let x $\in X$ - point from dataset. Then $z_1 = \mathbb{ENC}(x)$ - compressed representation of the data, and $z_2 = \mathbb{ENC}(\tau(x))$ - compressed representation of the transformed data. Concatenated vector $z = [z_1, z_2]$ fitted into the $\mathbb{DEC}$ and output of the decoder will predict the parameters of transformation τ. The visualization of this process plotted on the Figure 2.

The training process now depends on how to measure "the distance" between transformations. The distance will be some function $l(\tau, \hat{\tau})$. Then parameters of models will be found via procedure:

$$\hat{\theta}_E,\ \hat{\theta}_D = \text{argmin}_{\theta_E,\ \theta_D}\, \mathbb{E}_{x\in X,\ \tau \in T}\ \{l(\tau(x), \hat{\tau}(x))\}$$

3.3 Transformations

Depending on the metric and analytic form of transformation we can construct different losses, that can include parameters of transformation as well. Exist two types of transformations:

1. Parametrized transformations

   Family of transformations can be defined via some parameters.

   $$\text{T} = \{\tau_\theta | \theta \in \xi\}, \text{ where } \xi \text{ is the distribution of reachable parameters}$$

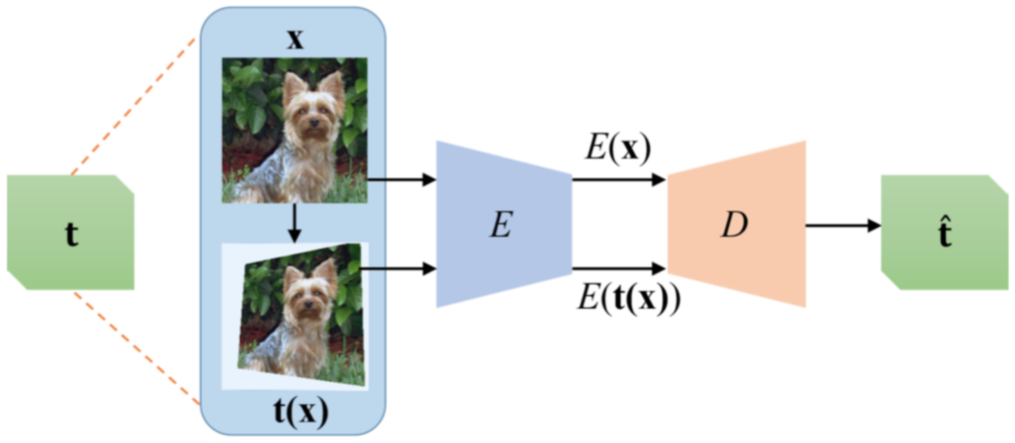

Figure 2: Encoder-Decoder scheme of transformations [8]

In such case, the parameters of distribution determine the entire transformation, so we can use it to build proper loss function between transformations

$$l(\tau_{\theta 1}, \hat{\tau}_{\theta 2}) = ||\theta_1 - \theta_2||_2$$

Authors of paper [8] used the affine and projective transformations (homography) as the distribution of possible transformations. They can be represented via matrix $M(\theta)$ with shape 3×3, this matrix translates one coordinate system (original image) into another (resulted image). Affine transformation of image can be viewed as translating all pixels to the new space with new coordinates:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \text{ where}$$

$\theta_A = \{a_{11},\ a_{12},\ a_{21},\ a_{22}\}$ - parameters of rotation and scale, $\theta_B = \{b_1,\ b_2\}$ - parameters of translation, i.e bias of coordinate system. In our experiments we used only affine transformations.

2. Non-parametric transformations

In the case if transformation very hard to parametrize we can construct non-parametric loss between transformation and its approximation as mean difference between transformations over all data samples:

$$L(\tau,\ \hat{\tau}) = \mathbb{E}_{x \in X}\ l(\tau(x),\ \hat{\tau}(x))$$

Authors of paper suggest non-parametrized loss isn't good enough in their experiments, because the performance on resulted vectors representations wasn't good as vectors trained with the parametrized loss. Hence, in our experiments we used only parametrized transformations (affine) and these type of functions that lies beyond this report.

3.4 Decoder-encoder architecture

In our experiments, we used authors Network-in-Network architecture - BlockNet - and evaluated performance on the CIFAR10 dataset. The encoder part consists of many several blocks with different types of layers. Basic block is sequence of repeating convolution, batch normalization and activations - ReLU. Below is demonstrated the example of block from the middle of the architecture.

```
BasicBlock(
    (layers): Sequential(
        (Conv): Conv2d(96, 192,
                       kernel_size=(5, 5), stride=(1, 1),
                       padding=(2, 2), bias=False)
        (BatchNorm): BatchNorm2d(192,
                       eps=1e-05, momentum=0.1,
                       affine=True, track_running_stats=True)
        (ReLU): ReLU(inplace)
      )
    )
```

Also, we replaced the Network-in-Network architecture with adapted AlexNet architecture and train the representations with help of this architecture.

The AlexNet was trained for feature representation learning using stochastic SGD algorithm with learning rate equals to 0.01 (dynamically changes over time depending on current epoch) and Nesterov momentum decay. The same metaparameters and optimizator used for BlockNet architecture.

The detailed information of parameters and architecture structure can be found in our GitHub with code and experiments [10]

3.5 Classification architecture

As features from classification used outputs from the middle of the neural network. As classsifier, we used the sequence of linear layers with batch normalization and ReLU:

```
Classifier(
    (Classifier): Sequential(
        Flatten()
        Linear(in_features=1536, out_features=512, bias=False),
        BatchNorm1d(512, eps=1e-05, momentum=0.1,
                    affine=True,
                    track_running_stats=True)
        ReLU(inplace)
        Linear(in_features=512, out_features=256, bias=False)
        BatchNorm1d(256, eps=1e-05, momentum=0.1,
                    affine=True,
                    track_running_stats=True)
        ReLU(inplace)
```

```
            Linear(in_features=256, out_features=10, bias=True)
            Softmax()
        )
    )
```

The classificators were trained on the top of pretrained models learning stochastic SGD algorithm and ADAM optimize, with learning rate equals to 0.01 (dynamically changes over time depending on current epoch) and Nesterov momentum decay (for SGD only). The accuracy on the test set with classificator on the top of BlockNet features is 0.9.

4 Experiments

In our experiments, we tested algorithms on the open dataset CIFAR10 [9]. CIFAR-10 dataset consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. For model building and training we used PyTorch, for similarity search we used scikit-learn and FAISS, and for plotting vectors in 2-dim space we used UMAP. For example, Figure 3 shows the result of the similarity search for few random chosen images from CIFAR10 using AET features.

More experiments (similirity search on vectors obtained from both methods and visualization of embeddings in 2D space) you can find at our GitHub in the notebooks folder [10].

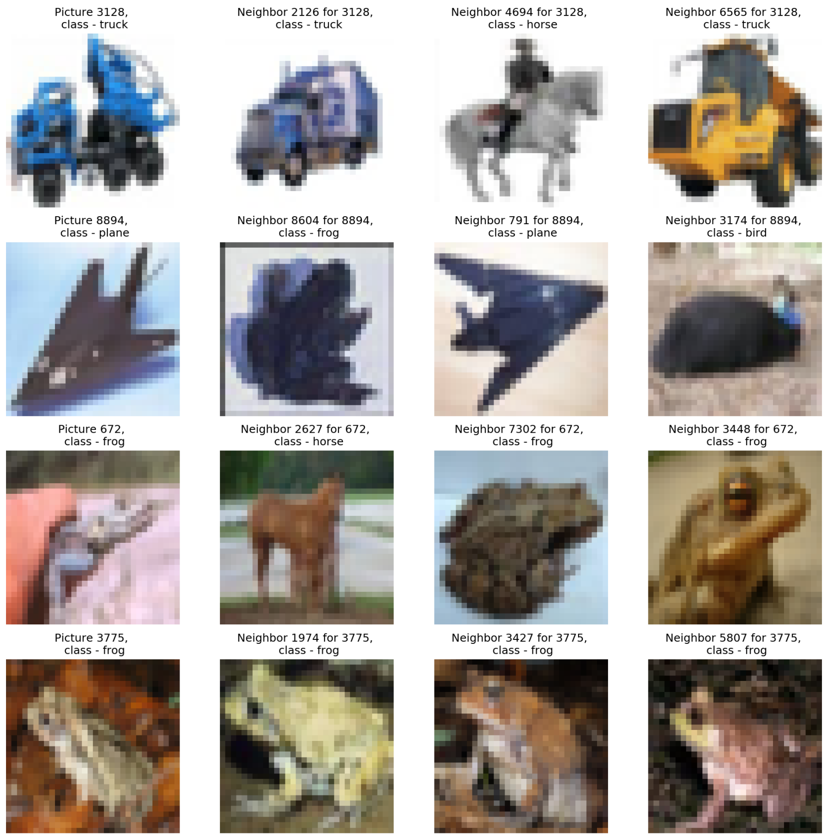
Picture 3128, class - truck
Neighbor 2126 for 3128, class - truck
Neighbor 4694 for 3128, class - horse
Neighbor 6565 for 3128, class - truck
Picture 8894, class - plane
Neighbor 8604 for 8894, class - frog
Neighbor 791 for 8894, class - plane
Neighbor 3174 for 8894, class - bird
Picture 672, class - frog
Neighbor 2627 for 672, class - horse
Neighbor 7302 for 672, class - frog
Neighbor 3448 for 672, class - frog
Picture 3775, class - frog
Neighbor 1974 for 3775, class - frog
Neighbor 3427 for 3775, class - frog
Neighbor 5807 for 3775, class - frog

Figure 3: AET method's similarity visualization

5 Conclusion & Future work

During this project we explored two recent papers on unsupervised learning, which becomes more and more important topic of the research. We got hands-on experience working with methods proposed in those papers and acquired better understanding of the entire field. However, we found out that performance of such methods is still far from the desired. For example, they don't capture semantic information, which is crucial for classification task. Also, we understand that there is no good single metric to estimate representations and how to measure their quality is an open question, which we want to investigate deeper in the future.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun.
Deep Residual Learning for Image Recognition [URL]

[2] Alec Radford, Luke Metz, Soumith Chintala.
Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks [URL]

[3] Spyros Gidaris, Praveer Singh, Nikos Komodakis.
Unsupervised Representation Learning by Predicting Image Rotations [URL]

[4] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, Yoshua Bengio
Learning deep representations by mutual information estimation and maximization [URL]

[5] Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm.
Mine: mutual information neural estimation [URL]

[6] M.D Donsker and S.R.S Varadhan.
Asymptotic evaluation of certain markov process expectations for large time, 1983

[7] Laboratory of Machine Perception and Learning [URL]

[8] Liheng Zhang, Guo-Jun Qi, Liqiang Wang, Jiebo Luo
AET vs. AED: Unsupervised Representation Learning by Auto-Encoding Transformations rather than Data [URL]

[9] Alex Krizhevsky, Vinod Nair, Geoffrey Hinton
CIFAR10 Dataset [URL]

[10] GitHub with experiments and evaluation [URL]