## Introduction

Recently, I have started to experiment with self-supervised learning techniques. In the last few months, we can see the rapid development of the methods. Moreover, self-supervised learning is closing the gap reaching transfer learning performance.

This family of methods allows using of unlabeled data to increase the performance of the task (e.g. classification, detection or so on). To better understand the idea behind this family of methods visit https://arxiv.org/pdf/1902.06162.pdf or https://github.com/jason718/awesome-self-supervised-learning. Self-supervised learning is used not only in computer vision domain, here is the example of successful use of self-supervised learning in automatic speech recognition https://ai.facebook.com/blog/wav2vec-state-of-the-art-speech-recognition-through-self-supervision/

In this post, I focus on the paper pretext invariant (...)

## Idea

Self-supervised learning usually involves preparing a pretext task. For instance, apply rotation to the images, then train the network to predict the rotation angles. Thus, this pretext task becomes a simple classification task. The advantage of this approach is that we do not need to have manually assigned labels. We can generate labels automatically. The aim of the pretext task training is to obtain good quality image representations that can be useful in pretext tasks. So, after pretext task pretraining, we can train our network on a downstream task (our target task). This approach is similar to transfer learning that also involves two stages – training on big dataset then training on a target task.

In the paper I describe in this post, authors also perform data augmentation. However, they focus not on the prediction of the properties of data augmentation transformation. Instead, they try to make the representation of the original image and a modified image similar. This makes sense. For example, the image of the dog should produce the same feature vector regardless of the rotation angle. This can lead to more robust computer vision algorithms.

(photo)

How to make the network produce the same representations? By proper design of the loss function.

So the idea is to pass the original image and modified image by the network. Then change the network parameters in the optimization process to make the representation returned from both two vectors the same. It can be obtained by minimizing the distance between those two representations. Unfortunately, there is one problem. The network can learn the trivial solution, for example, return zeros for all examples. In order to avoid such solutions, the distance between modified image and other imags in the dataset should be maximized. This lead to the loss function of the following form:

d(o,m) / suma(d(o,m)

Note, that this function is similar to softmax function. Therefore, we can use cross-entropy loss function available in deep learning libraries.

You can notice the representations in delimiter, obtaining it every batch could be very costly, so they decided to construct a representation bank that contains representations of all images in the dataset. So instead of calculating that every time we can look up the array.

They slightly modified the loss function. Firstly, the compare representations from memory to representation of original image, that dampen the change of parameters. Moreover, modified loss allows for making original image and unmodified image from the database not similar.

They made an extensive analysis of many tasks. I'm not going to describe this here. What is interesting, the authors managed to beat the object detection task against transfer learning.

## Implementation

I decided to implement the algorithm in the Pytorch library. The code is available here: https://github.com/akwasigroch. I performed initial experiments on the ISIC 2017 dataset. The dataset contains 2000 training images, 150 validation images and 600 test images. I obtained the following results on ResNet50 network:

- 0.53 AUC - training from scratch (the network can't learn)

- 0.71 AUC - training using a self-supervised pre-trained network

- 0.81 AUC - training using transfer learning from a network trained on Imagenet dataset

I did not manage to reach the performance of transfer learning. However, the results are promising, and I think an increase in the number of images (I used the same images in supervised and self-supervised training) can lead to much better results. This is especially important in medical cases when there are many images that are not labeled, as it is time-consuming and involves the work of a skilled medical specialist.