VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning

VICReg Application to the Diabetic Retinopathy Analysis Challenge 2022

Seunghee Jeong
seunghee.jeong@campus.lmu.de

Abstract

This report first introduces the concept of self-supervised learning and one of the famous methods, Variance-Invariance-Covariance Regularization (VICReg) for self-supervised learning. And shows how VICReg can perform well on self-supervised learning through initial implementation to a different dataset, CIFAR-10. Finally, by implementing the Diabetic Retinopathy Image classification task hosted by Diabetic Retinopathy Analysis Challenge (DRAC22) and with its results, we will see how it affects performance in a real-world application.

1 Introduction

In the real world, it is not full of labels, actually, even though we have some labels but lack labels with datasets. Self-Supervised Learning (SSL) is a method of machine learning which learns from unlabeled sample data. Therefore, it can be one of the good solutions. In recent years, SSL has produced promising results and has found practical application in audio processing and is being used by Facebook and others for speech recognition [4]. Nevertheless, achieving high accuracy is still challenging. Variance-Invariance-Covariance Regularization (VICReg) is one of the state-of-the-art methods for self-supervised learning. It results on par with the state of the art on several downstream tasks, such as SimCLR, MoCo v2, SwAV, BYOL, etc, as it mentioned in the paper [1].

1.1 Self-Supervised Learning (SSL)

Self-Supervised Learning (SSL) is a Machine Learning paradigm where a model, when fed with unstructured data as input which is unlabeled, generates data labels automatically, which are further used in subsequent iterations as ground truths [5]. Hence, it can be regarded as an intermediate form between supervised and unsupervised learning base on artificial neural network (ANN). The primary appeal of SSL is that training can occur with data of lower quality, rather than improving ultimate outcomes. SSL more closely imitates the way humans learn to classify objects [4].

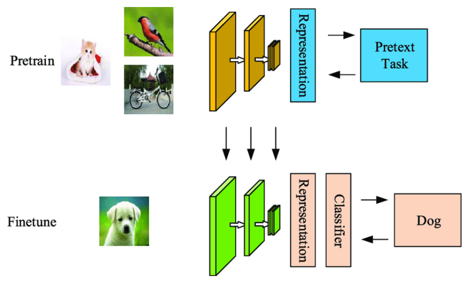
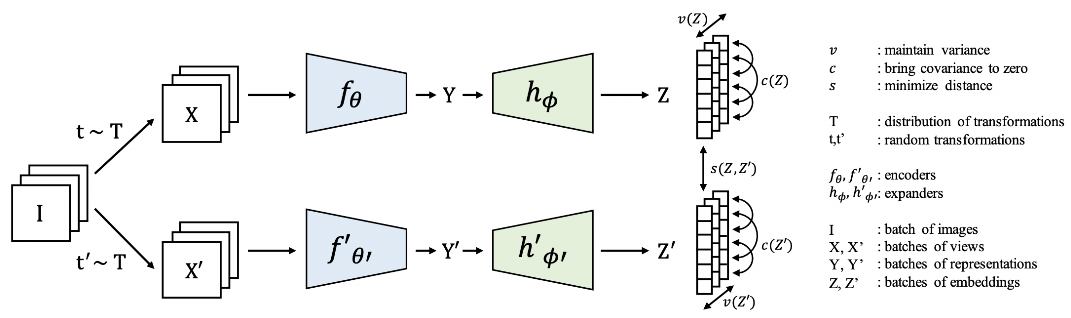


Figure 1: Self-Supervised Learning

1.2 VICReg: Variance-Invariance-Covariance Regularization

A self-supervised method for training joint embedding architectures based on the principle of preserving the information content of the embeddings [1]. The basic idea is to use a loss function, which is composed with three terms, variance, invariance and covariance.



$v(Z)$
$c(Z)$
$s(Z,Z')$
$c(Z')$
$v(Z')$
I
X
X'
t ~ T
t' ~ T
f_θ
$f'_{\theta'}$
Y
Y'
h_ϕ
$h'_{\phi'}$
Z
Z'
v : maintain variance
c : bring covariance to zero
s : minimize distance
T : distribution of transformations
t,t' : random transformations
$f_\theta, f'_{\theta'}$: encoders
$h_\phi, h'_{\phi'}$: expanders
I : batch of images
X, X' : batches of views
Y, Y' : batches of representations
Z, Z' : batches of embeddings

Figure 2: VICReg: joint embedding architecture with variance, invariance and covariance regularization.

1.2.1 Variance-Invariance-Covariance Regularization Loss Function

$$\ell\left(Z, Z'\right)=\lambda s\left(Z, Z'\right)+\mu\left[v(Z)+v\left(Z'\right)\right]+\nu\left[c(Z)+c\left(Z'\right)\right] \tag{1}$$

The variance, invariance and covariance terms that compose VICReg loss function.

1. **Variance**: constrain the variance of embedded vector along each dimension independently
2. **Invariance**: make the embedding from different image view close to each other
3. **Covariance**: prevent the network encode similar information to the same dimension in the embedded space

1.2.2 VICReg Loss Function with Three Terms

1. **Invariance**: the mean square distance between the embedding vectors.

$$s\left(Z, Z'\right)=\frac{1}{n} \sum_{i}\left\|z_{i}-z_{i}'\right\|_{2}^{2} \tag{2}$$

2. **Variance**: a hinge loss to maintain the standard deviation (over a batch) of each variable of the embedding above a given threshold. This term forces the embedding vectors of samples within a batch to be different.

$$v(Z)=\frac{1}{d} \sum_{j=1}^{d} \max \left(0, \gamma-S\left(z^{j}, \epsilon\right)\right) \tag{3}$$

$$S(x, \epsilon)=\sqrt{Var(x)+\epsilon} \tag{4}$$

3. **Covariance**: a term that attracts the covariances (over a batch) between every pair of (centered) embedding variables towards zero. This term decorrelates the variables of each embedding and prevents an informational collapse in which the variables would vary together or be highly correlated.

$$C(Z)=\frac{1}{n-1} \sum_{i=1}^{n}\left(z_{i}-\bar{z}\right)\left(z_{i}-\bar{z}\right)^{T}, \quad where \bar{z}=\frac{1}{n} \sum_{i=1}^{n} z_{i} \tag{5}$$

$$c(Z)=\frac{1}{d} \sum_{i \neq j}[C(Z)]_{i, j}^{2} \tag{6}$$

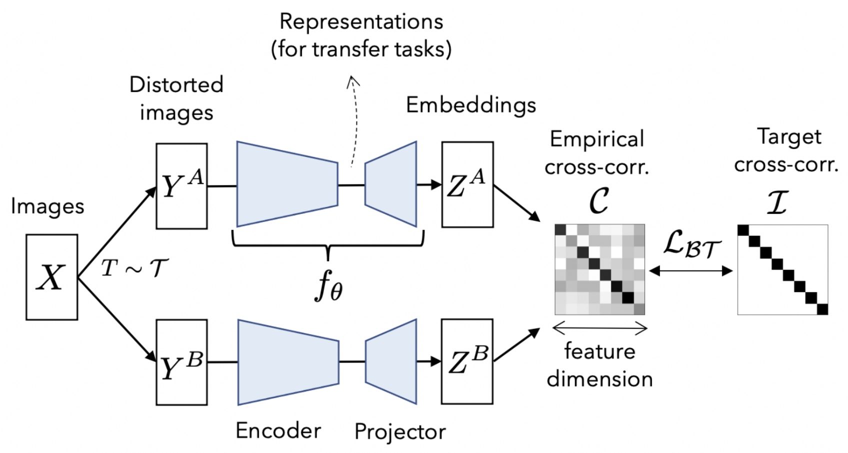


Figure 3: Barlow Twin Objective

2 Initial Implementation

Initial implementation of VICReg on CIFAR-10 dataset is applied to two-part, with Layer-wise Adaptive Rate Scaling (LARS) Optimizer and Stochastic Gradient Descent(SGD) Optimizer. Due to the feature of each optimizer, comparing the results of both optimizers showed different effects on loss values.

2.1 With LARS Optimizer

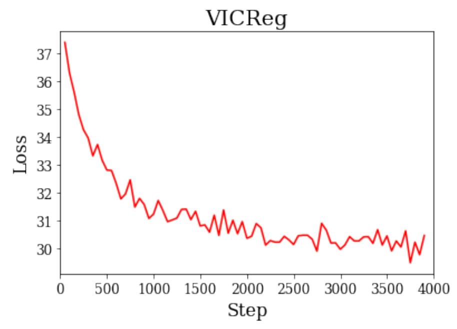
Layer-wise Adaptive Rate Scaling(LARS) **Layer-wise Adaptive Rate Scaling, or LARS**, is a large batch optimization technique. There are two notable differences between LARS and other adaptive algorithms such as Adam or RMSProp [2]:

1. LARS uses a separate learning rate for each layer and not for each weight.
2. The magnitude of the update is controlled with respect to the weight norm for better control of training speed.

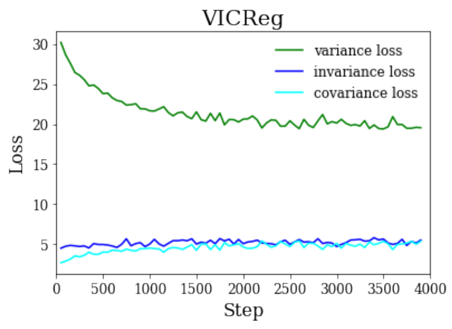
$$\begin{gathered} m_t = \beta_1 m_{t-1} + (1-\beta_1)\left(g_t + \lambda x_t\right) \\ x_{t+1}^{(i)} = x_t^{(i)} - \eta_t \frac{\phi\left(\left\|x_t^{(i)}\right\|\right)}{\left\|m_t^{(i)}\right\|} m_t^{(i)} \end{gathered} \tag{7}$$

Model Setting

- Backbone: ResNet-18
- Optimizer: **LARS (Layer-wise Adaptive Rate Scaling)**
- Loss: **VICReg loss**
- Hyperparameters
  - λ=μ = 25, ν = 1
  - mlp hidden dim = 2048
  - batch size = 256
  - **learning rate = 0.3**



(a) VICReg Loss



(b) VICReg Loss with Three Terms

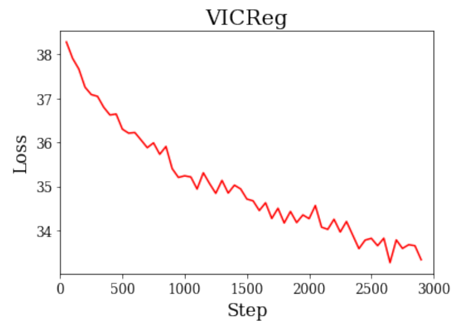
Figure 4: VICReg Loss with LARS Optimizer and Loss with Three Terms

2.2 With SGD Optimizer

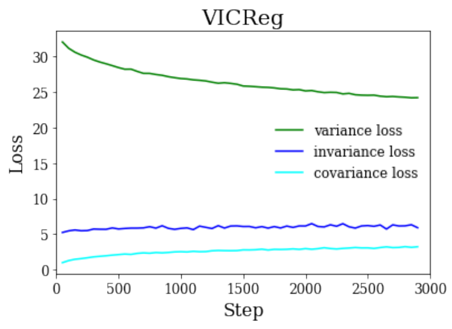
Since LARS is more suitable for the large batch size, changed to sgd optimizer with smaller learning rate.

Model Setting

- Backbone: ResNet-18
- Optimizer: **SGD**
- Loss: **VICReg loss**
- Hyperparameters
  - λ=μ = 25, ν = 1
  - MLP hidden dim = 2048
  - batch size = 256
  - **learning rate = 0.001**



(a) VICReg Loss



(b) VICReg Loss with Three Terms

Figure 5: VICReg Loss with SGD Optimizer and Loss with Three Terms

Compared to LARS, SGD gives less noisy loss value. However, in general, LARS gave better loss values than SGD's loss.

3 Introduction of the DRAC22: Diabetic Retinopathy Analysis Challenge

The challenge DRAC22 (Diabetic Retinopathy Analysis Challenge 2022) is a first edition associated with MICCAI 2022. Three tasks are proposed (participants can choose to participate in one or all three tasks):

- Task 1: Segmentation of Diabetic Retinopathy Lesions
- Task 2: Image Quality Assessment
- **Task 3: Diabetic Retinopathy Grading**
  - Image classification task by the grading
  - Around 600 images and 3 multi-classes

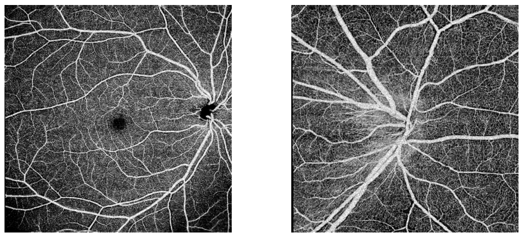


Figure 6: Dataset Image Samples for Diabetic Retinopathy Grading

4 Implementation and Result

4.1 Challenges and Blockers

- Complicated code structures and highly bound to each other
  - Very customized code structure was difficult to understand and not easy to change and simply add new codes
  - Not intuitive and complex code to make universal for many models
  - Split to so many minor functions and they are bound so complicated
  - Had to understand all the connection between codes and different files
- Challenge dataset size is very small
  - Total around 600 images so even smaller image dataset for training
  - Easy to be overfitting and difficult to increase validation accuracy
- Very careful to select hyperparameter setting
- Training and Evaluation of these setting was very time consuming
- Validation accuracy doesn't improve

4.2 Application for Improvement

Parameter tuning Depending on the initial parameter setting, the model can be effected a lot. Tried to find better combination of hyper parameters by differing the parameters values, e.g. **learning rate**, **model complexity** like hidden neurons or number of hidden layers, **batch size** and other also miner tuning. Since the dataset size is small, reducing the model capacity and batch size. The model was really sensitive to the learning rate, the model was overfitted when the learning rate is larger than 0.02 and when the batch size was larger than 64. As much as the each hyperparameter could effect the model, the combination of those parameter were also impact on the model. Therefore, the model could learn in the right way by conducting the learning with these combinations many times.

Optimizer Initial optimizer was lars optimizer and by changing to sgd optimizer. In order to avoid overfitting and also speed up, **adam optimizer** is used for final implementation.

Early Stopping Setting the **patience** parameter to 5 and **monitoring** parameter to VICReg loss so the model could stop when its VICReg loss doesn't improve more than 5 steps, it stops learning.

Weight Decay Since the size of the dataset is small, using weight decay can help overfitting problem.

Resize the image size Original size of image dataset is 1024×1024. By resizing the image size to 224×224 and 512×512 and compared to the original size one, 1024×1024. Finally, 224×224 one performed the best, therefore, resize parameter is 224.

4.3 Results

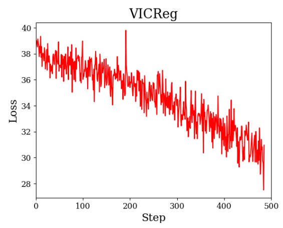
Model Setting Since LARS optimizer is more suitable for the large batch size, changed it to sgd optimizer and used smaller learning rate.

- Backbone: ResNet-18
- Optimizer: **Adam**
- Loss: **VICReg loss**
- Hyperparameters
  - λ=μ = 25,ν = 1
  - MLP hidden dim = 128
  - batch size = 32
  - weight decay = 1e-4
  - learning rate = 0.015

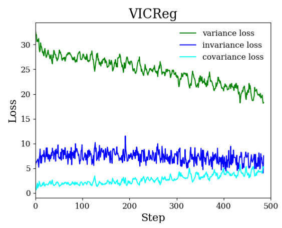
4.3.1 Results: Fine Tuning with Fully-Connected Layers

$\Rightarrow$ **Overfitting**: Test accuracy didn't really improve and very unstables

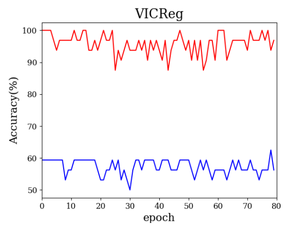
Reason for the Overfitting Problem and Possible Solutions For Transfer Learning, if the size of the dataset is small, usually the pre-trained model is big and it can induce overfitting so as in our case. In this case, the pre-trained model size might be too big for the small dataset. (e.g. many hidden layers, big size of hidden neurons). Therefore, simpering classifier models will help. Also adding regularization techniques (e.g. Lasso, L2-regularization, Dropout, Weigh decay, Early stopping, etc) will be helpful. Over-sampling to make the sample size bigger, and changing various parameters can be also other solutions.



(a) VICReg Loss



(b) VICReg Loss with Three Terms

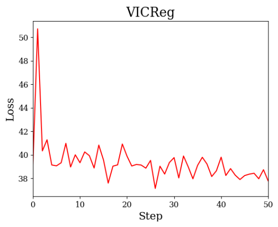


(c) Train Accuracy and Test Accuracy

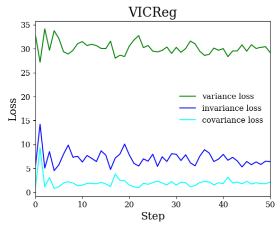
Figure 7: Transfer Learning before Improvement

4.3.2 Results: Fine Tuning with Convolutional Layer + Fully-Connected Layers

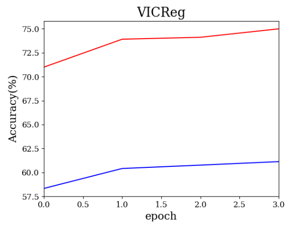
⇒ **Test accuracy shows slightly increasing phase** The result after fine-tuning shows a slight improvement than before applying the solutions. The reason why it could be better was simpering the model capacity by keeping hidden layers as small as possible, decreasing the size of hidden neurons, as well as tuning the using the weight decay, early stopping, parameter tuning, differently resizing of the images, reducing the batch size, and using suitable optimizer for it.



(a) VICReg Loss(Blue-Test, Red-Train)



(b) VICReg Loss with Three Terms



(c) Train Accuracy and Test Accuracy

Figure 8: Transfer Learning after Improvement

5 Conclusion

The goal of this project is the Image classification application of one of the state-of-the-art self-supervised methods, VICReg to the challenge dataset. To conduct this project, I faced two main challenges.

At first, the VICReg codes were so customized so I had to understand the whole structure of the codes and created the custom dataset and adapted data loader following their codes. There were completely new class objects, 'DatasetBase', and also had to be familiar with torch lightening and all.

Secondly, transfer learning for a small datasets can often easily induce overfitting problems, so it needs proper solutions based on a precise diagnosis for the problem and needs to learn through lots of trials and error. It was the main and the hardest challenge of this project. By application of several methods for improvement, I could alleviate the problem of the previous model and improve the model. In order to simper the model capacity, I kept hidden layers as small as possible and decreased the size of hidden neurons. Moreover, by tuning the using the weight decay, early stopping, parameter tuning, and differently resizing of the images. Since the size of the dataset is not enough to have a big batch size, reduced the batch size, and used a more suitable optimizer for it. Consequently, I could get some improvements as below:

1. The model became slightly stabler than before
2. Alleviation of overfitting problem
3. Test accuracy showed increasing phase

Furthermore, the model still needs to be stabilized and to perform better. Accordingly, for future work can be further parameter tuning, further fine-tuning, still using other regularization methods, trying with a larger learning rate, and over-sampling. Hopefully, selecting a different pre-trained model which is more specialized for medical images might be helpful, at least worth trying. For example, it might give different or even better results. Because when the pre-trained models are being trained in the pre-training stage, each model would learn representation learning differently affected by transformation methods, training image features, and so on. Therefore, their performances cannot be the same, that is, there might be out-performing pre-trained models especially suitable for medical images. Finally, Interestingly, there is one paper dealing with transfer learning with a small dataset. The paper suggests targeted transfer learning method, called 'cycle training', discriminative learning rates with gradual freezing and parameter modification after transfer learning, which gives the best results among all experiments [3]. There will be more possible works and applying these solutions will be expected to handle the facing overfitting problem of transfer learning in small datasets better.

References

[1] Adrien Bardes, Jean Ponce, and Yann LeCun. *VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning*. 2021. DOI: 10.48550/ARXIV.2105.04906. URL: https://arxiv.org/abs/2105.04906.

[2] *LARS*. https://paperswithcode.com/method/lars.

[3] Miguel Romero et al. "Targeted transfer learning to improve performance in small medical physics datasets". In: *Medical Physics* 47.12 (Oct. 2020), pp. 6246–6256. DOI: 10.1002/mp.14507. URL: https://doi.org/10.1002%2Fmp.14507.

[4] *Self-Supervised Learning*. https://en.wikipedia.org/wiki/Self-supervised_learning.

[5] *The Beginner's Guide to Self-Supervised Learning*. https://www.v7labs.com/blog/self-supervised-learning-guide#h1.