Self and Semi-Supervised Deep Learning

Somesh K. Singh, Rohit Garg, Aditya Mishra
CS F425[Deep Learning]: Major Project
Department of Comp. Sc., BITS Pilani, K.K.Birla, Goa Campus
NH 17B, Bypass, Road, Zuarinagar, Sancoale, Goa 403726

Abstract

This work compares the performances of three models: ConvNet, ResNet18 and our own WRN-45-2 through various experiments. For all tasks, we use STL-10 dataset exclusively. We do not use any pre-trained weights in any of our model architectures. In this work, we will be exploring supervised deep learning approach, constrastive self-supervised learning using Barlow Twins loss and t-SNE visualizations for study of embeddings, semi-supervised deep learning approach using pseudo labelling technique. This work contains ablation study of varying ratios of labelled and unlabelled data on all three models. This work produces an increment in accuracy of prediction of WRN-45-2 from 72 %(supervised) to 75 %(semi-supervised). It is inferred that semi-supervised approach produces better performance than both of individual supervised and self-supervised approach. This project is done in partial fulfillment of course CS F425[Deep Learning] under the guidance of Prof. Tirtharaj Dash.

Introduction

Supervised learning approaches are very good at learning from labelled data. However, obtaining labels is often resource intensive. There is an abundance of unlabelled data available on the internet. In this work, we tend to answer two questions: 1. Are there ways in which we can learn from this vast resource? 2. Does 'lack of labels for data imply redundant learning?

This is where the idea of learning representations comes in. The specific type of self-supervised learning we will be exploring is Contrastive Unsupervised Learning, which is based on a simple idea - representations of similar input data should be similar and representations of dissimilar data should be dissimilar. Since our representations will be real valued vectors in z $\in$ Rn, similarity can be simply thought of as distance under some metric between the vector. On answering first question, we further need to overcome two challenges: 1 a. Finding similarity in unlabelled data. 1 b. Learning representation in absence of labels. In this work, we tackle the first challenge by using data augmentation techniques to create multiple versions of the same image. All of these versions (from the same image) are treated as similar when training the model to produce appropriate representations. For an image x, one obtains two augmented versions x1 and x2. One obtain embeddings of these by passing them through the model you want to train: z1 = fθ(x1) and z2 = fθ(x2)). Our aim in contrastive representation learning is to make representations (or embeddings) of similar inputs similar so ideally z1 and z2 should be the same. We train our model to minimise an appropriate loss L(z1, z2) through gradient descent on θ. Different approaches use different losses and also add particularities to make this general pipeline work better. In this work, we use Barlow Twins our contrastive self-supervised representation learning loss. The goal is to keep the representation vectors of augmented versions of one batch similar, while minimizing the redundancy between them.

We have used Adam optimiser for our experiments. All our experiments were performed using Google Colab.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Architecture

In this work, we will be using three architectures.

- ConvNet (baseline)
- ResNet 18
- WRN-45-2

We do not use any pre-trained weights in any architecture. ConvNet architecture is our baseline, Convolutional Neural Network model. It consists of 4 BasicConv blocks followed by nn.Flatten layer. Each BasicConv block is essentially convolutional layer with kernel of size 3x3 and stride of 2 followed by ReLU activation layer. Dropout is also applied to regularise the network and control overfitting. When a batch of images with batch size of b is fed to network, an embedding of dimension $(b, 512)$ is produced during forward propogation. An overview of ConvNet architecture is given in Figure 1.

In this work, ResNet 18 architecture (He et al. 2016) is used as a reference for comparision of performance with randomly initialised weights and biases. Final fully connected layer of ResNet 18 was replaced with nn.Flatten layer in order to produce an output tensor of dimension $(b, 512)$.

Motivated by (Zagoruyko and Komodakis 2016), we build our own WRN-45-2 having 45 convolutional layers and k = 2 widening factor. Leaky ReLU activation was used instead of ReLU activation. Compared to ResNet architecture, the order of batch normalisation, activation and convolutional in residual block was changed from Conv-BN-ReLU to BN-LeakyReLU-Conv. Our WRN-45-2 consists of initial Conv2d followed by 4 Network Blocks. This is followed by BatchNorm2d, LeakyReLU activation layer, Average Pool 2d layer and flattening layer in the same order.

Each Network Block consists of 5 Basic Blocks basic0, basic1, basic2, basic3 and basic4. The basic0 block consists of 2 series of arrangements of BN-LeakyReLU-Conv. There is a skip connection from first LeakyReLU activation to output layer of basic0 via Conv2d layer. The basic1, basic2, basic3 and basic4 blocks all share the same 2 series of arrangements of BN-LeakyReLU-Conv and there is an identity skip connection from input layer to output layer.

WRN-45-2 produces an output tensor of dimension $(b, 128)$.

```
ConvNet(
  (conv): Sequential(
    (0): BasicConv(
      (conv): Conv2d(3, 128, kernel_size=(3, 3), stride=(2, 2))
      (relu): ReLU()
      (dropout): Dropout2d(p=0.0, inplace=False)
    )
    (1): BasicConv(
      (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2))
      (relu): ReLU()
      (dropout): Dropout2d(p=0.0, inplace=False)
    )
    (2): BasicConv(
      (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2))
      (relu): ReLU()
      (dropout): Dropout2d(p=0.0, inplace=False)
    )
    (3): BasicConv(
      (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2))
      (relu): ReLU()
      (dropout): Dropout2d(p=0.0, inplace=False)
    )
    (4): BasicConv(
      (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2))
      (relu): ReLU()
      (dropout): Dropout2d(p=0.0, inplace=False)
    )
    (5): Flatten(start_dim=1, end_dim=-1)
  )
)
```

Figure 1: ConvNet Architecture (baseline)

```
(1): BasicBlock(
  (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.001, affine=True, track_running_stats=True)
  (relu1): LeakyReLU(negative_slope=0.1, inplace=True)
  (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.001, affine=True, track_running_stats=True)
  (relu2): LeakyReLU(negative_slope=0.1, inplace=True)
  (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
)
```

Figure 2: Basic Block unit in WRN-45-2

Dataset

In this work, we will be using the STL-10 dataset (Coates, Ng, and Lee 2011), an image recognition dataset inspired by ImageNet dataset with some improvements. It consists of two parts. The labeled part consists of ten classes of images with 500 training and 800 test images per class. The unlabelled part consists of 100000 unlabeled images. There are ten classes in STL10 dataset: airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck. All the images are 3 color channel images with 96×96 pixels in size.
Test set was further split into validation and test set in 1:4 ratio.

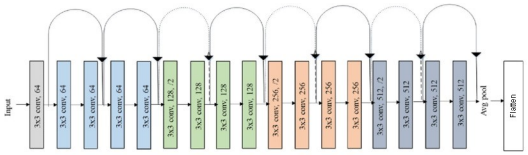

Figure 3: ResNet Architecture

Table 1: Results for supervised learning

Model	Train Accuracy	Val. Accuracy	Train Loss	Test Accuracy
ConvNet	48.8	59.36	1.41	59.58
ResNet18	68	71.4	0.86	71.68
WRN-45-2	74.1	76.1	0.73	73.18

Supervised Deep Learning

Implementation Details: Learning rate was initialised to $2 * 10^{-3}$ and the training was performed for 100 epochs with checkpoint after each epoch. Each image from training set is augemented by applying random Zoom, Horizontal Flip, Color Jitter, Gray scaling and Gaussian Blur. We then normalise the image using mean and standard deviation of the dataset. In each epoch, a batch of 125 images are fed to train the network. We apply a Linear layer on top of each of the model architectures. The output of this layer is a 2d tensor of dimension $(batch\_size, 10)$ representing probabilities of each label for the batch of images. Since there are 10 classes, we use cross entropy loss as our supervised loss metric.

Results: Upon training, train accuracy, train loss and validation accuracy for ConvNet, ResNet18 and WRN-45-2 are shown in Table 1. We infer that the order of learning performance of models is WRN-45-2 >ResNet18 >ConvNet.
During model evaluation on test dataset, the above trend is followed and test accuracies are in same order. This is evident from Table 1.

Self-Supervised Deep Learning

The goal of self-supervised learning is to learn useful representations of the input dataset.

Augmentation: Each input image is transformed twice to produce the two transformed views. The image augmentation pipeline consists of the following : random zooming, horizontal flipping, color jittering, converting to grayscale, Gaussian blurring, and solarization. The first two transformations (cropping and resizing) are always applied, while the last five are applied randomly, with some probability. This probability is different for the two augmented versions

Table 2: Results for self-supervised learning

Model	Train Accuracy	Validation Accuracy	Train Loss
ConvNet	40.11	42.98	23.7
ResNet18	38.27	39.45	28.88
WRN-45-2	42.94	44.95	16.05

in the last two transformations (blurring and solarization).

Barlow Twins Method Barlow Twins is a constrastive loss function for representation learning. It's objective function measures the cross correlation matrix between the embeddings of two identical networks fed with augmented versions of a batch of samples, and tries to make this matrix close to the identity (Zbontar et al. 2021).

Implementation Details: The encoder consists of model architecture (without the final classification layer) as described above. Two randomly augmented versions (y_a, y_b) of batch of unlabelled images are fed to this encoder network to produce two embeddings (z_a, z_b). We then compute the barlow twins loss between these two embeddings. For this experiment, the learning rate was initialised to $5 * 10^{-4}$ and we train our model for 4 epochs.

Results:

Upon training, train accuracy, train loss and validation accuracy for ConvNet, ResNet18 and WRN-45-2 are shown in Table 2. We infer that the order of learning performance of models is WRN-45-2 >ResNet18 >ConvNet.

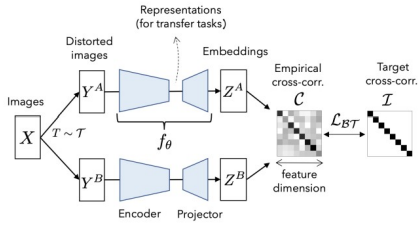

Figure 4: Barlow Twins Loss Pipeline. Adapted from work of (Zbontar et al. 2021)

Embeddings & Visualisation

During self-supervised model training, we compute TSNE visualisations from embeddings. (t-SNE)t-Distributed Stochastic Neighbor Embedding (van der Maaten and Hinton 2008) is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data (Pedregosa et al. 2011).

Observations: A detailed inspection of the t-SNE map reveals that much of the local structure of the data (such as the orientation of the ones) is captured as well. Among all three models, WRN-45-2 classifies the unlabelled dataset into most accurate categories. ConvNet classifies data into two dominant categories(vehicles and animals). ResNet18 provides a better classification of categories. However, the distinction between categories is still lower than WRN-45-2. After 4 epochs of self-supervised training, all 3 models learn higher-level feature representations from unlabelled data and seperation between the clusters become more prominent. This is evident from t-SNE visualization of WRN-45-2 during self-supervised learning, as described in Figure 5.

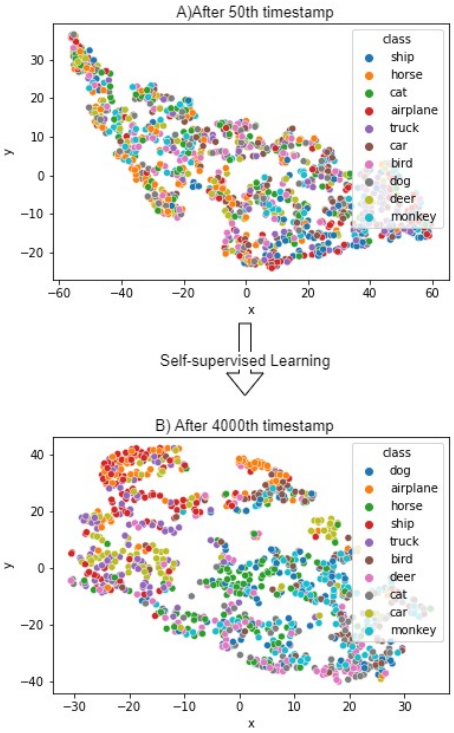

Figure 5: t-SNE visualization of evolution of feature representation space during self-supervised learning of WRN-45-2. A detailed inspection of the t-SNE map reveals that much of the local structure of the data (such as the orientation of the ones) is captured as well and seperation between the clusters become more prominent with training.

Semi-Supervised Deep Learning

Our Semi-supervised approach is inspired by Google Research's Fix-Match approach (Sohn et al. 2020). We use consistency regularisation (minimising the gap between augmentations of same image) and pseudo-label (making use of model's predictive power to label unlabelled data). However, we do not apply a strict threshold after creating pseudo-labels. We also observe that Adam optimizer gives

better convergence and performance than the orginial SGD implementation.

Implementation Details: A supervised model is trained on the labeled images and corresponding cross-entropy loss is computed from a batch of 32 images per epoch. Two augmented versions, strong and weak are generated for each unlabeled image from batch of 64 images per epoch. Our model receive weakly augmented image and we get prediction over classes. Henceforth, the strongly augmented image is passed and predicted class labels are obtained. Cross-Entropy is used to compare this probability distribution with ground truth pseudo-labels. Both the supervised and self-supervised losses are combined and the model is tuned as shown in Figure 6. In this way, we combine both the supervised and self-supervised training procedures to make our model more robust.

Data Augmentation: Our implementation leverages two kinds of augmentations: "strong" and "weak". In both augmentaions, we apply randomized flip, re-cropping, jitter, grayscaling, Gaussian Blur, Solarization, normalization. However, strong and weak augmentations differ in the probabilities with which these above functions are applied.

Results: Upon training, train accuracy, train loss and validation accuracy for ConvNet, ResNet18 and WRN-45-2 are shown in Table 1. We infer that the order of learning performance of models is WRN-45-2 >ResNet18 >ConvNet.

During model evaluation on test dataset, the above trend is followed and test accuracies are in same order. This is evident from Table 3. We also experimented using the self-supervised model weights as a starting point. However, it yields no improvement.

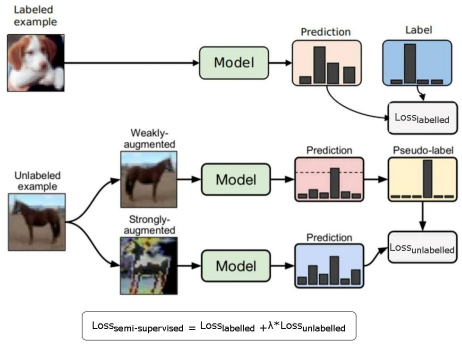

Figure 6: Semi-Supervised Pipeline

Ablation Study

Ablation studies on various ratios of lablled and unlabelled data: We compare the performance of all three model architectures using semi-supervised learning approach by varying ratio of labelled and unlabelled data given as input. Ratios taken are 1:1, 1:5 and 1:20. It can be observed that as amount of unlabelled data is increased, both training accuracy and validation accuracy increase in general. The change is significant in ConvNet and marginal in ResNet18 and WRN-45-2 as shown in Table 4.

Table 3: Results for semi-supervised learning

Model	Train Accuracy	Val. Accuracy	Train Loss	Test Accuracy
ConvNet	50	56.65	1.96	55.63
ResNet18	79.32	73.77	0.61	72.45
WRN-45-2	87.75	75.2	0.25	75.17

Table 4: Ablation study results

Ratio	ConvNet		ResNet18		WRN-45-2	
	Train	Val.	Train.	Val.	Train.	Val.
1:1	49.68	55.61	73.71	72.1	74.18	74.05
1:5	54.78	58.06	76.43	73.2	75.04	74.71
1:20	54.86	58.72	81.2	75.55	82.18	76.08

Inference & Discussions

WRN-45-2 performs better than ResNet and ConvNet due to the fact that in ResNet, the gradient flows through skip-connections instead of residual connections. This leads to loss of feature representation learning in the network and less information retention. This problem was formulated as diminishing feature reuse in (Srivastava, Greff, and Schmidhuber 2015). Due to its widened Network Blocks, WRN-45-2 does not reuse diminished features.
It can be inferred that in our setup, semi-supervised learning approach produces much better training and validation accuracies as compared to supervised and self-supervised learning approaches due its combined loss function.

References

Coates, A.; Ng, A.; and Lee, H. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 215–223. JMLR Workshop and Conference Proceedings.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Sohn, K.; Berthelot, D.; Li, C.-L.; Zhang, Z.; Carlini, N.; Cubuk, E. D.; Kurakin, A.; Zhang, H.; and Raffel, C. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015.

Highway networks. corr abs/1505.00387. *URL http://arxiv.org/abs/1505.00387*.

van der Maaten, L., and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9:2579–2605.

Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zbontar, J.; Jing, L.; Misra, I.; LeCun, Y.; and Deny, S. 2021. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*.