RnD Project Report
Self Supervised Transfer Learning and Kaggle Challenge Application

Amey Patil - 160050006
Computer Science and Engineering IIT Bombay
ameypatil@cse.iitb.ac.in
Guide: Prof. Amit Sethi

Abstract

The developments in Deep Learning over the time has provided major breakthroughs in ability to comprehend complex datasets with deeper models in Computer Vision and Healthcare AI. But the the basic requirement to achieve the optimal performance of any Deep Learning system is a large good quality dataset to train the deep CNNs. And hence the trade-off between the data size/quality and performance exists. The current primitive solutions to use ImageNet pretrained model is restrictive in terms of the context of dataset. To tackle these problems, we aexplore self-supervision based approaches on two different datasets that first performs self-supervision on data and then is used as initialization to fine-tune on target task. The code can be found here: Kaggle Challenge and Jigsaw

Introduction

In computer vision, convolutional neural networks (CNNs) have demonstrated tremendous power in understanding images. CNNs have shown close to human level annotator performance in natural image settings on a wide variety of tasks and datasets. The success of deep learning was due to the large amounts of curated data fed to deep networks for training. To learn high dimensional representations, large and 'good' datasets are required. But in most of the real world problems, for example radiology, these conditions hardly hold. Performance of deep learning models deteriorate in low supervision scenarios. The conventional way to deal with the medium sized dataset is to use Imagenet pretrained weights [1] and fine-tune them on the target task. This is called Transfer Learning and is widely used for almost every task with CNN networks. The main idea behind using the imagenet pretrained weights is to avoid random initialisation of weights. Convolutional Neural Networks progressively extract complex features from the image and the final dense layer uses this extracted features to learn a decision boundary. The first few layers of the CNN architecture extract basic features such and edges, corners and shapes. The next few CNN layers learn to extract the more and more complex features and hence can learn the data distributions.

Using ImageNet weights relies on the fact the ImageNet trained features are generic and can be used to fine-tune on any task and achieve good performance. This may be the thumb rule for the most of the tasks, but this approach has its limitation as not every dataset has a similar distribution as Imagenet dataset, which consist of natural images of 1000 objects. For example, Radiograph Diagnosis datasets have a very different distribution and also have very subtle features seperating the image classes. Hence this approach is not reliable when we have small size dataset and the data distribution is complex and the model needs to learn more complex features. In contrast fine tuning the architecture over weakly extracted features can be more hard given final dense layers has to solve for a more tough decision boundary. Radiology is a very good example to demonstrate this limitation. A typical radiograph is a grayscale image, and the datasets are more noisy and representing a complex data distribution with high variance. So the motivation for exploring a dataset focused parameter initialisation comes forward. In this work we explore Jigsaw Pretext as a initial task for pretraining. The main idea is to solve a secondary task on the same dataset and in the process learn basic data distribution to get a good initialisation parameters for the main task.

Datasets

1. Radiology Diagnosis Dataset - CheXpert

This is high resolution Chest Radiograph dataset [4] labelled for multilabel classification task. The dataset has 14 different pathology classes including a No Finding class. The dataset itself is very large consisting of over 100,000 radiographs. For the purpose of this project, to simulate the limited data size prominently, we use only 10% of the im-

ages from the dataset chosen at random without disturbing the class distribution. We use all 14 classes for the training and testing purposes and treat this as a multilabel classification problem. For evaluation we follow the metrics posed by CheXpert challenge as the micro AUC score for the top 5 findings: (a) Atelectasis, (b) Cardiomegaly, (c) Consolidation, (d) Edema, and (e) Pleural Effusion.

2. Plant Pathology Detection Dataset - Kaggle

The dataset is taken from an active Kaggle Competition "Plant Pathology 2020 - FGVC7" [5] and consists of very high quality plant leaf images with 4 categories. The idea behind choosing this dataset is to show the limitation of imagenet pretrining as well as the trade-offs of image augmentation techniques. The data has 1800 development images and hence can be considered as a very small dataset. For evaluation we use the hidden test datasset on the Kaggle Platform.

Jigsaw Problem

In this method we take the deep CNN model and solve the Jigsaw pretext problem as a secondary task, and use the trained weights to fine-tune them on the main task. The system pipeline has two components, i) the self-supervision pretext task and ii) the radiology target task. The complete system setup is demonstrated in Figure [1].

1. Introduction

The procedure of the Jigsaw puzzle pretext task is similar to that explained in [2]. We have illustrated the self-supervision jigsaw task in Fig. 6.

The image is partitioned into 9 equal sized tiles. From these tiles, patches are extracted and feed to the CNN network *H*. The patches are shuffled using a randomly chosen arrangement from a predefined set of 1000 arrangements. Let this predefined set of arrangements be $M : Z \rightarrow Z^9$ such that $M(i)$ represents a permutation of 9 locations $\{1, 2, 3, ...9\}$ for each $i = 1, 2, ..1000$. This means if the input $tiles_1$ are shuffled according to a particular permutation $M(i)$, then the network output should be a 1000 element one hot vector activated only at position i.

Considering all permutations of tiles becomes infeasible ($9! = 362880$), a suggestion is to use 1000 permutations with sufficiently large average hamming distance. We pick patches such that a random gap of 0 to 22 pixels is maintained between neighbouring patches. All these techniques prevent undesirable solutions like learning low level statistics and trivial solutions from edge continuity.

2. Experiments

The encoder architecture or the Context-Free Network is used to encode images into feature representations. We consider fully convolutional networks as encoder architectures. We choose VGG19 encoder as the context-free network. VGG19 has fewer parameters making to easier to avoid overfitting on datasets. DenseNet121 although having lower number of parameters is a 121 layer deep convolutional network with skip connections, making to slower and difficult to train. We show our results on both VGG19 and DenseNet121 models.

1) **Training self-supervision**: From each image of size 256x256, a section of size 225x225 is randomly taken. This ensures the focus not being just in the center crop but in other parts of the image as well. This section is partitioned into 9 tiles each of size 75x75. Further, from each of these tiles a random patch of size 64x64 is taken. This ensures that there is a random gap between 0 and 22 pixels maintained between patches. Finally they are shuffled according to randomly chosen permutations from a list of 1000 per-

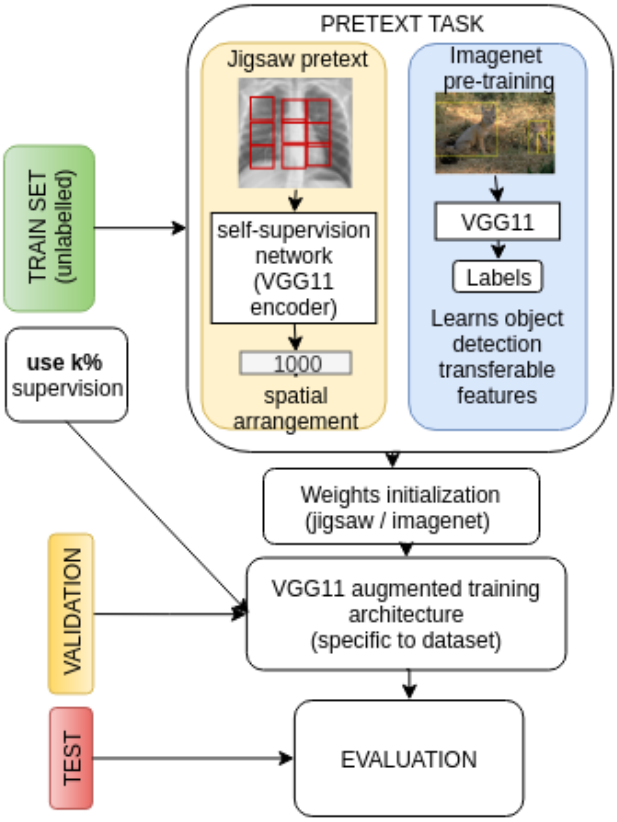


Figure 1: First using the complete training set (unlabelled) the self-supervision network learns structure in the data via jigsaw puzzles. An alternative initialization by Imagenet pre-training on the same network is used for comparison. The target task network encoder is initialized using the pretext task. This is fine-tuned on target task with 10% labels from training set and performance is evaluated on test set.

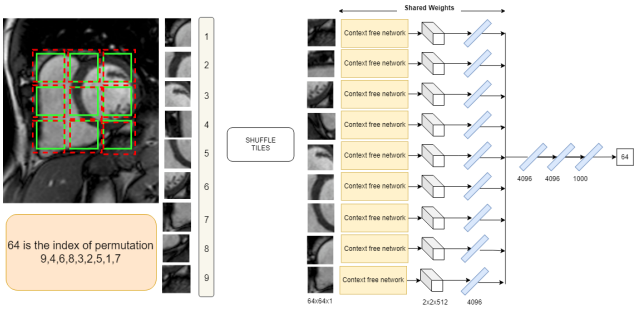


Figure 2: Jigsaw puzzle self-supervision task is shown. An image is partitioned in 9 tiles shown by red dotted lines. Random patches are extracted from each tiles, totalling 9. According to a permutation randomly chosen, patches are shuffled and fed to the CNN. All layers (including the CFN) have shared weighted before concatenation. The inference is an index which was used from the mapping to shuffle the patches. In this format the network learns the spatial orientations which leads to a meaningful image. Output dimensions are shown below each layer. The output of CFN is a feature volume, as shown. Layers in blue are fully connected layers.

mutations. This is fed to the siamese-ennead network for training. All experiments use cross-entropy loss. The system is run for 200 epochs with SGD learning $rate = 1e-3$ with $10x$ decay every 200 epochs. To avoid large weight updates, L2 $penalty = 1e-4$ is imposed on weights. We train self-supervision on the whole training set.
2) **Training on target task**: On top of the trained Context-Free Network from self-supervision training, we attach an appropriate task head network depending on whether classification needs to be performed. This is the preparation of network T as mentioned in algorithm . Next, we input whole images to the network T and with the help of corresponding groundtruth, we fine-tune T using cross entropy loss. Class balancing in the loss is applied wherever applicable. In all our experiments, all layers are kept trainable. We use Adam optimizer for training. A dropout factor of 0.8 is used for the classification network. Slight rotation and shifts were commonly used data augmentations. All these hyperparameters were tuned on validation set to prevent overfitting and achieve best performance. As we see further that our experiments are based on less supervision, overfitting was a major concern and was tackled with these measures.

The Jigsaw task is a easy classification task the final accuracy is saturated at 0.99 on test set.

3. Results

3.1 Radiology Diagnosis

We demonstrate 2 settings of the experiment for 10% and 100% of the CheXpert dataset. The chosen metric is AUC score for the top 5 labels as specified in the CheXpert challenge [4]

Model	Dataset	Pretraining	AUC
VGG-19	10% CheXpert	ImageNet	0.65
VGG-19	10% CheXpert	Jigsaw	**0.68**
Densenet-121	10% CheXpert	ImageNet	0.63
Densenet-121	10% CheXpert	Jigsaw	**0.64**
VGG-19	100% CheXpert	ImageNet	0.84
VGG-19	100% CheXpert	Jigsaw	**0.85**
Densenet-121	100% CheXpert	ImageNet	**0.87**
Densenet-121	100% CheXpert	Jigsaw	0.86

Table 1: AUC scores on CheXpert Validation set.

4. Conclusion

For the smaller size of the dataset the advantage of using the Jigsaw pretraining is evident over using ImageNet weights. Also, the lower AUC values for the smaller set are a clear indication that Deep Learning model are data hungry and the dataset size impacts hugely to the performance of the model. The full dataset training setup leads to a very good accuracy. The effect of Jigsaw pretraining is masked in this case as we are using the full 100,000 image sized dataset. Also, because of the lower parameter count for VGG network the network performs better for 10% data split but as the dataset size is increased, the deeper densenet beats the VGG network.

Kaggle Challenge

The active Kaggle Challenge - Plant Pathology 2020, is a part of FGVC7 programme. The objective of the challegne is to develop robust models to classify the hidden test plant leaf images to the given four categories: 1) healthy, 2) multiple diseases, 3) rust, 4) scab.

1. Introduction

The dataset has a very high quality RGB images of close pictures of plant leaves. The main challenge posed is to use the very limited size dataset to train very deep CNN models. Hence an ideal setting for Transfer Learning explorations. We have run a numerous experiments over the baseline model. The evaluation criterion is average AUC score provided by the Kaggle hidden test set.

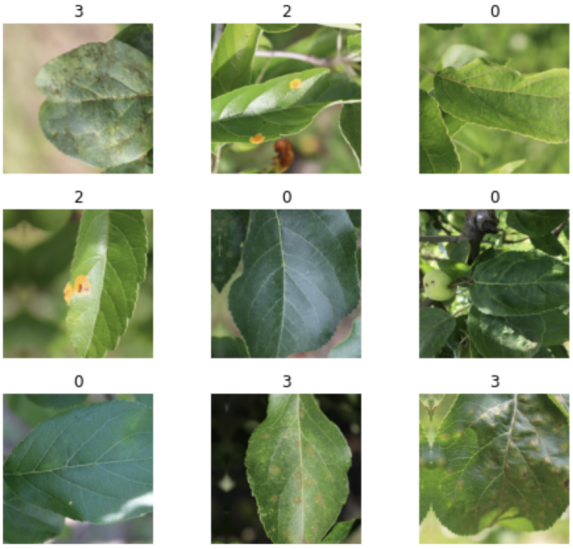


Figure 3: The development images sampled with 4 possible labels:- 0:healthy, 1:multiple-diseases, 2:rust, 3:scab

2. Experiments

We have run numerous experiments on the Kaggle Platform and have submitted 58 submissions to evaluation. For the comparison we have chosen the best performances for Densenet121 model with Jigsaw and ImageNet pretraining. Also, we have used an external dataset PlantVillage [6]. Also we demonstrate the use of extensive image augmentations to componsate for smaller dataset size and avoiding overfitting. The dataset consists of very large sized images(2000x1300), but in some images only a smaller portion of the image has the target leaf. Hence for efficiently using the target information from the image, we have used a cropped dataset using a pretrained leaf detection algorithm [7] and fine-tuned the weights over only the cropped dataset. Finally we use a very deep and wide network EfficientNet-B7 to compare the results on the hidden test set. We use a custom learning rate scheduling as shown in the Fig. 4 and found the initial lower learning rate to be useful. We have used basic image augmentation such as: 1) horizontal, verticle flip, 2) shift 0.05xsize, 3) Random rotation (-15, +15) in all the experiments. For the Ext. Augmentation experiments, we have used ImageNet pretrained weights and added more augmentation as 1) Gaussian noise, 2) bilateral filtering, 3) shear, 4) scaling, 5) center cropping(0.9).

3. Results

For smaller image size, 320x320, the difference in the performances for the pretraining is prominently evident.

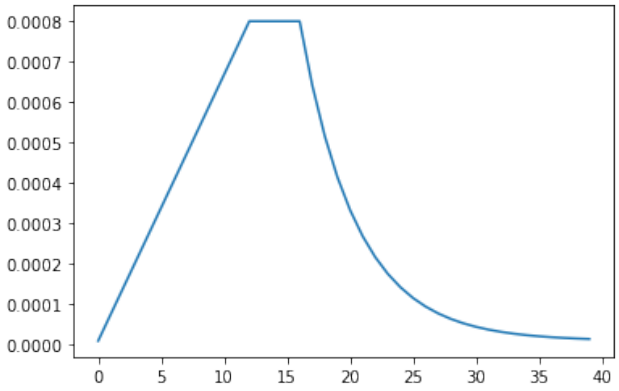


Figure 4: Learning Rate Scheduler

The AUC score for Jigsaw pretrained model is greater for both 320 and 768 image size. Also it is interesting to note that the Jigsaw advantage over the ImageNet pretraining is almost equivalent to using the external PlantVillage dataset for both image sizes. Also the Extensive Augmentations added to the dataset can damage the quality of the images and hence the final performance on the target task.

Model	Image Size	Pretraining	AUC
Densenet121	320x320	ImageNet	0.9462
Densenet121	320x320	Jigsaw	**0.9609**
Densenet121	320x320	PlantVillage	**0.9615**
Densenet121	320x320	Ext. Augment	0.9317
Densenet121	768x768	ImageNet	0.9713
Densenet121	768x768	Jigsaw	**0.9729**
Densenet121	768x768	PlantVillage	**0.9730**
Densenet121	768x768	Ext. Augment	0.9652
5-Fold Densenet	320x320	ImageNet	0.9538
5-Fold Densenet	320x320	Jigsaw	**0.9683**
5-Fold Densenet	320x320	PlantVillage	**0.9701**
EfficientNet-B7	768x768	ImageNet	**0.9735**
EfficientNet-B7	768x768	PlantVillage	**0.9831**
EfficientNet-B7	768-crop	ImageNet	0.9644
EfficientNet-B7	768-crop	PlantVillage	0.9648

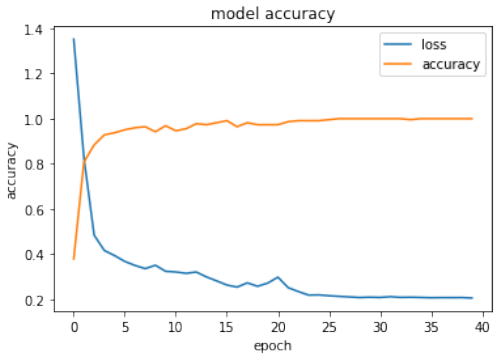
Table 2: All Performances

As the development data is limited we use 20% split for the validation during training. A very good approach to use all the data and get an ensemble is to train the models on K folds and average the final predictions at the inference time. The 5-Fold performance increases over the single model.

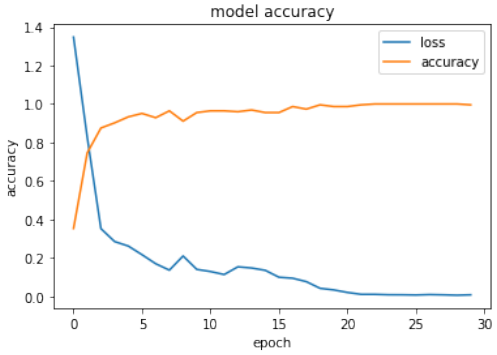
Finally we as we have very high quality dataset, using deeper and wider network, EfficientNet-B7 [3] helps the performance by far. The Jigsaw pretraining was not feasi-

ble for this model as the model FCN depends on the image size and we can not use it for the slices crops during solving Jigsaw pretext. Using PlantVillage pretrained weights as initialisation helps the perfoemance to reach the best single model AUC - **0.983**. Note that the EfficientNet being more sophisticated model beats the densenet models, the comparison of the different pretraining settings must be done with the densenet experiments only.

The secondary advantage of Jigsaw pretraining is the early saturation of the model while training. In Fig 5, the Jigsaw pretrained model reaches the saturation at epoch 15, whereas the ImageNet pretrined model takes more than 20 epochs. This is a very useful advantage in computation required, also the probability that the model overfits to train data decraeases over longer training sequence.



(a) ImageNet



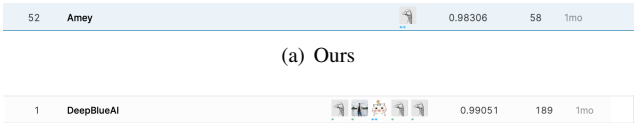
(b) Jigsaw

Figure 5: Validation loss and Accuracy: Quicker saturation for Jigsaw pretrained network lowers the risk for model overfitting compared to ImageNet pretraining.

Conclusion

We show that for more complex problems like Radiology Diagnosis Transfer Learning using Jigsaw weights is very useful over the traditional ImageNet weights.

We have made 58 submissions to the Kaggle challenge. The project successfully explores the problem of the scarcity of dataset for CNN networks. The final selected submission ranks **52nd** on the Public Leaderboard with AUC score **0.98306**, against the first place submission with AUC score **0.99051**.



(a) Ours

(b) First position

Figure 6: Standing on Kaggle Public Leaderboard with AUC score as the metric.

Future Work

In this work, we only explore the Unsupervised learning method for pretraining task, and we are not using the full resources at hand given a labelled dataset. A future extension to this work can be training a feature extraction module just like we trained using Jigsaw setup, but by solving a Label Clustering problems. The Siamese Network like encoder architecture can be trained on the dataset to cluset the data first. This can be more useful and more dataset oriented as while solving clustering problem the model can learn features vital to the classification problem and hence pose a very easy problems for the target task while fine-tuning the parameters.

References

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks

[2] M. Noroozi and P. Favaro, Unsupervised Learning of Visual Representations by solving Jigsaw puzzles

[3] Mingxing Tan ,Quoc V. Le EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

[4] CheXpert: A Large Chest X-Ray Dataset And Competition

[5] Plant Pathology 2020 - FGVC7 `https://www.kaggle.com/c/plant-pathology-2020-fgvc7/overview`

[6] PlantVillage Dataset - Plant Disease Detection `https://github.com/spMohanty/PlantVillage-Dataset`

[7] Cropped Plant Pathology Dataset - Plant Disease Detection
https://www.kaggle.com/inseltiger/my-first-data/settings