Transfer Learning - A Deep Learning Approach

David Ye davidye@kth.se
Felix Meli felixnm@kth.se
Alexander Gutell agutell@kth.se
Ludvig Skare lskare@kth.se

Abstract

Deep learning models have helped us in accomplishing many feats in modern society. Although state-of-the-art models such as ChatGPT, DALL-E and VALL-E are the best in their fields, they are also computationally expensive and require an abundant amount of resources. One solution is to make use of transfer learning, a method that utilizes a pre-trained model as a starting point to solve a similar task. Through this method, fewer resources are required and can also improve generalization.

Similarly to many other deep learning approaches, transfer learning requires labeled data. The process of manually labeling data is both time-consuming and labor-intensive. In the last decade, however, semi-supervised learning has become a hot topic to relieve the need for labeled data. This method of training models trains on labeled data and exploits unlabeled data to further improve learning performance. One such method is known as pseudo-labeling. This method has gained popularity and attention in recent times.

This report analyzes an implementation of transfer learning on Oxford's pet dataset. In addition, a thorough analysis of semi-supervised learning using pseudo-labeling as an extension to this project will be performed. The final test accuracy on the multi-classification task was 93.5% which suggests the model classifies the classes well. In comparison to other existing models, it seems visual transformers perform better than ResNets in this task. The results from semi-supervised learning indicate that pseudo-labeling improves the performance if the amount of labeled data is limited but more experiments and other variants of this approach should be further test

1 Introduction

In the last few years, deep learning has proved to be a success in many fields and garnered attention from all around the world [3]. The seemingly infinite potential of deep learning and the consideration it accumulated never ceased. There is however one big problem with developing deep learning models - they require loads of resources. The development and training of deep learning models is a tedious and time-consuming task. Moreover, a lack of data will affect the trained models and the performance will be worse than anticipated. One approach to solving the need for loads of data is to implement transfer learning. Transfer learning is a deep learning method where a model developed to solve a task is reused for solving a similar task. The benefits of using transfer learning include less data needed, faster training and better generalization performance.

Even though less data is needed, they still need to be labeled for the supervised models to work. In reality, data is often manually labeled which is a tedious process. This raises an important question - how can both datasets be utilized to train deep learning models? The answer lies in semi-supervised learning. Weng proposes several approaches for implementing semi-supervised learning in her report [11]. One of them includes *pseudo labeling*, a method that assigns fake labels to unlabeled data. The fake labels are given according to the highest softmax probabilities predicted by the model.

Subsequently, the combined datasets will be used to train the previous model to improve accuracy and generalization.

In order to explore the effect of transfer learning and the usefulness of pseudo labeling, we will conduct experiments using transfer learning as well as different ways of implementing pseudo labeling and analyze the results. The transfer learning model was used to classify a binary and multi-classification task, although only the results of the latter task is showed due to the binary task being a trivial task in comparison to the multi-classification task.

2 Related Work

The published work by fastai [7] has implemented a similar model using the same dataset. In contrast to our work, fastai uses fastai library instead of PyTorch and also uses an in-built function that automatically finds the best learning rate. This automatic detection will yield a better learning rate than our coarse and fine lambda searches. The related work also uses ResNet34, which is different from our ResNet152. Furthermore, their data augmentation differs a little where they resize the images to a larger size before batching.

Apart from performance-related differences, fastai extracts labels using regular expressions by looking at the filename. According to fastai, this is a common approach for extracting labels since they are usually filenames. This was not required by us since we were given all labels at the beginning. However, in a similar fashion to using regular expressions, we needed to separate the classes from each other and used the labels to put them in their respective folders.

3 Data

The Oxford-IIIT Pet Dataset which the model was trained on is given by the University of Oxford [8]. It consists of 37 classes of dog and cat species with roughly 200 .jpg images for each class. The images vary in their properties including scale, pose and lighting. In addition to the images, ground truth labels for the respective image as well as tight bounding box (ROI) for their heads and pixel level trimap segmentation were included. However, the trimaps have not been used in this project.

Most of the best-performing state-of-the-art models use a visual transformer (ViT) network to classify the dataset [1]. They obtain an accuracy of over 95% which is a significant increase over ResNet-152-SAM which has 93.3% accuracy. The visual transformer models perform better than the ResNet used in this report. Wang et al. [4] noticed that, unlike ResNet, ViTs have extremely sparse active neurons for the first layers. A more sparse representation of data encourages feature extraction which results in better generalization.

4 Methods and experiments

In order to implement transfer learning, a pre-trained model had to be utilized. The pre-trained model ResNet152 was used and implemented by following PyTorch's guide [2]. Multiple pre-trained models including ResNet18, ResNet34 and ResNet50 were also tested but proved to perform worse. The final layer of the model was first replaced to solve the binary classification task, which ended with a test accuracy of 99.51% and subsequently replaced to solve the multi-class classification task. The splitting of the dataset was partly conducted by us. We were given 50% of the dataset as test data and could choose freely how to split the remaining 50% into training and validation data. We followed a standard approach to split it into 80/20.

It should also be mentioned that an NVIDIA T4 was used in this project. However, due to the GPU's limitations, our batch size was set to 4. A higher value would cause multi-threading issues.

4.1 Multi-class classification

The multi-class classification task is more advanced than binary classification. Several finetuning methods were therefore tested including finetuning layers, different learning rates, data augmentation and batch-norm.

For the finetuning experiment, we tried to vary between activating different layers. Figure 3 shows combinations of finetuning different layers. Finetuning other layers did not seem to have a huge impact but the performance did improve a little. The best combination to finetune was layer4, layer2, layer1 and fully-connected layer.

ADAM optimizer with a learning rate of $\eta = 0.00005$ was used. This value was found through a coarse lambda search which can be seen in figure 1. It should be noted that this was the first chosen learning rate. After implementing other improvements, the learning rate had to be decreased to compensate for other changes in the model.

Data augmentation can be implemented in several ways. The purpose of using augmentation is to transform the images to improve generalization and prevent overfitting. In cases where data is sparse, one can augment the training data and add them to the training set. It should however be noted that too much augmentation on the same data may lower the performance. Our approach to augmenting the data was to crop a random section of each image and resize them to fit the input layer of ResNet. Moreover, a random horizontal flip was performed on the images. Other methods to obfuscate data such as rotating and changing the brightness were experimented with but they did not yield better performance. We also added extra augmented data to our training set but the performance worsened a little. In addition to these approaches, we also tried to utilize the ROI for their heads and use the head crops as training samples. This was completed by extracting the coordinates from the given XML files and cropping the corresponding images and resizing them. This approach did not yield better results either.

Lastly, unfreezing batch norms for different layers were also tested. By unfreezing batch norms, we enable a more stable network by circumventing the need for good weight initialization. This should result in a better overall performance. Figure 2 shows the result of iteratively adding layers to batch normalize. It is clear that having any kind of batch norm results in faster convergence. The final model ended with adopting batch normalization on all layers.

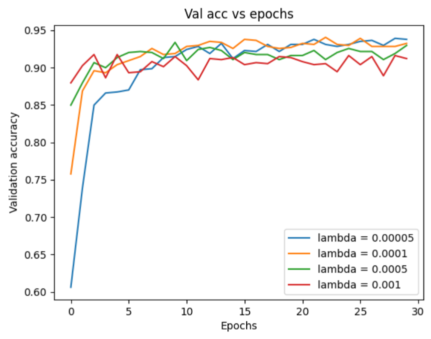

Figure 1: Different learning rate η for multi-class classification

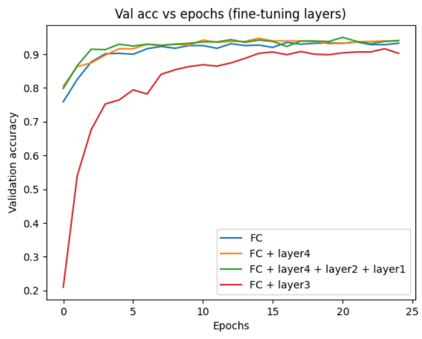

Figure 2: Finetuning layers for multi-class classification

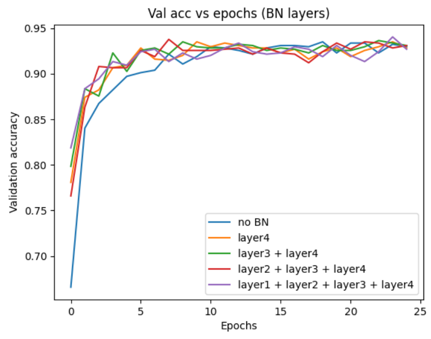

Figure 3: BN layers for multi-class classification

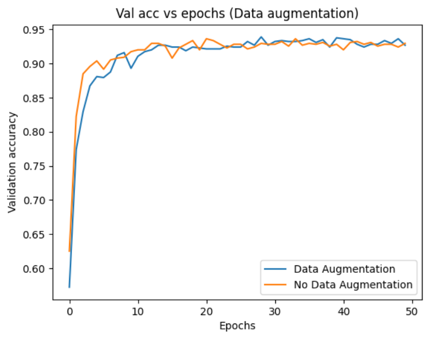

Figure 4: With/Without Data augmentation for multi-class classification

4.2 Pseudo-labeling

As mentioned earlier, pseudo-labeling is an approach to implement semi-supervised learning, mostly in order to produce decent to good accuracies even with less labeled data. There are multiple approaches to pseudo-labeling. In this project, we implemented different approaches but focused mainly on the self-training aspect of pseudo-labeling. This involves a "teacher" model that is trained on the labeled data, which evaluates and produces pseudo labels for the unlabeled data. This is followed by creating a "student" model, which is trained on the labeled and previously pseudo-labeled data combined. This process is optimally done in iterations where the previous student model becomes the new teacher. This way, in theory, the accuracy should improve somewhat after each iteration.

To make sure pseudo-labeling yields proper results and the student model performs better than the teacher (which is a needed criterion for the technique to work), there are multiple strategies that one should apply. It was found by Xie et al. [9] that not applying noise on the unlabeled data that the teacher evaluates results in better performance for the student. Through this implementation, the model can more easily produce correct classifications and thus provide better labels for the student model to train on. An important part of their experiments was to give the student model and its data noise in order to make the student perform better. In the article written by Xie et al. [9], random augmentations were applied to data as well as dropout and stochastic depth on the student. We ended up using random augmentations as well as random resize crops as noise for the data.

The teacher model was initialized with different amounts of labeled data. The original training data was first split into labeled and unlabeled data. We tested the pseudo-labeling method with 50/50, 10/90 and 1/99 splits for labeled and unlabeled data. One alternative method is to let the teacher model iteratively pseudo-label only a portion of the unlabeled data. The portion of the data which receives pseudo labels is the portion that the teacher is the most confident in its predictions. There are multiple ways to approach this, such as having a threshold for confidence in predictions or taking the top x predictions based on confidence. In this project, we compare labeling all together and the latter iterative alternative, as a way to consistently keep labeling more of the data, at the cost of sometimes having less accurate pseudo-labels. It should also be mentioned that at each iteration, the teacher had 5 epochs to train on the new dataset before labeling a new portion.

5 Discussion and results

5.1 Performance analysis

The final test accuracy resulted in 93.5%. Improvements to deciding learning rates could be improved. Since ResNet uses skip connections to remove vanishing gradients, the same value for the learning rate could be applied to all layers. However, we noticed that the layers converge at different speeds during fine-tuning. This could have caused the deviation in figure 3 where fine-tuning with fully connected and layer3 becomes slower. Layer3 also has the most parameters to tune, which would explain the unnatural behavior. Optimally, each layer should receive its own learning rate so that they all converge equally fast.

At first, the idea of adding more samples from data augmentation seemed to yield better performance. However, the results showed that the model did not perform better with added augmented samples. The most plausible explanation is simply that the model is overfitting on training samples, hence explaining why the model worsened a little after adding the samples. We were cautious about not applying too much augmentation and also experimented with not adding any augmentation at all. To prevent overfitting while adding more samples, we need to add new unseen samples that still resemble the different classes. This is further discussed in future work.

All of the above-mentioned possible improvement areas have contributed to making the model perform better. The learning rate is as usual an important factor in determining whether a model performs optimally or not. A bad learning rate often results in slow convergence or oscillations in accuracy and loss. The results from finetuning the layers and data augmentation have surprised us the most since they did not result in any significant improvement. Finetuning all layers is usually performed whenever the pre-trained model has trained on a dataset dissimilar to the new dataset. This can speed up the convergence rate and will result in better performance. Our experiments suggest that the pre-trained model is trained on a similar dataset since finetuning other layers only resulted in small improvements. Likewise, data augmentation only improved the model a little. This suggests that further improvement from any data-related changes has to come from adding new samples to the training dataset. The effect of batch normalization is also small but still contributed a little to improving the model.

5.2 Pseudo-labeling

When implementing pseudo-labeling of unlabeled data, several tests were conducted. One test was to compare an iterative learning approach and a labeling approach with a one-time labeling of all the unlabeled data. As seen in figures 5, 6 and 7, an iterative labeling approach performs equally well as a one-time labeling approach. The test accuracies in table 1 show the same result. In theory, however, the iterative approach should yield better results since this allows the model to slightly improve for each iteration. In turn, we receive a model that should be more accurate in labeling unknown data in the next iteration which results in a positive compound effect. Needless to say, it is difficult to evaluate whether an iteration approach performs better than no iteration.

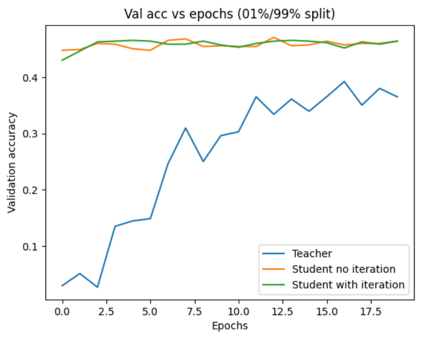

Figure 5: Validation accuracy with 1% labeled data for 20 epochs.

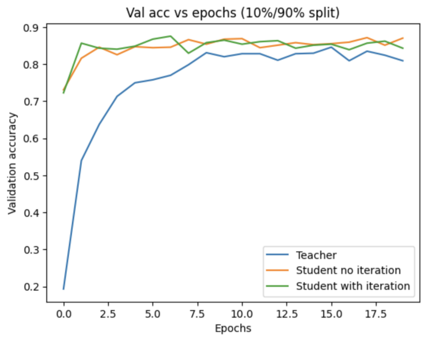

Figure 6: Validation accuracy with 10% labeled data for 20 epochs.

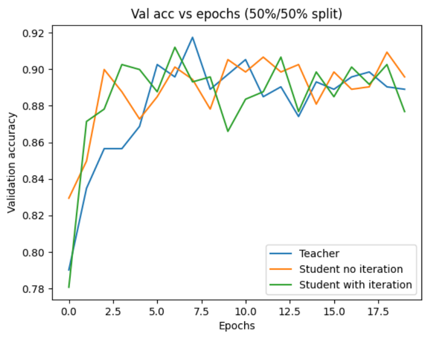

Figure 7: Validation accuracy with 50% labeled data for 20 epochs.

Another factor that was examined was the partitioning size of the labeled and unlabeled data. The results can be seen in figures 5, 6, 7 and 1. Naturally, more available data usually results in better performance. Thus using more labeled data leads to a better initial teacher, followed by a well-performing student trained on the good predictions of the teacher. One notable difference between the splits is that pseudo-labeling has a more positive impact on less available labeled data. This aligns with the theory of how the amount of data affects the models' performances. If the available data is small, then adding more data will improve the model more contrary to if the amount of available data was larger. Furthermore, the size of the provided labeled data to the teacher matters. In this case, we are working with approximately 3000 images for training but since the dataset is shuffled, we might not always receive an adequate amount of labeled data for each class. This shows that not only is it important to have a good ratio of labeled to unlabeled data, but also to have a decent amount of data to train the initial teacher so that the student can receive a meaningful amount of extra pseudo-labeled data to learn from.

The results also show that a much higher split of labeled data almost does not improve the performance. Although we do improve the model, the rise in accuracy may not always be significant depending on the balance of the amount of data and the complexity of the task.

Splits %	Teacher	Student without iterations	Student with iterations
1/99	0.4156	0.4691	0.4585
10/90	0.8193	0.8526	0.8681
50/50	0.8932	0.8860	0.89121

Table 1: Test accuracies for pseudo-labeling

6 Future work

There are several possible improvements to this work. As suggested by Chen et al. [4], experimenting with visual transformers would probably yield better results. The resulting ViT model obtained by Chen et al. outperformed ResNet without any large-scale pre-training or strong data augmentations. In combination with data from the top-performing models, ViTs currently perform better than ResNets.

It would be interesting to investigate how the performance would be affected with more training data. One alternative is to explore further which split between training and validation to choose. This could be achieved by performing n-cross-fold validation to assess the different splits. Another alternative would be to acquire new samples from generative models. Examples of generative models are Generative Adversarial Networks (GANs) or Variational Autoencoders (VAEs) which are highly known for their capability of generating new samples. Due to the limited data samples in this project, we would suggest adopting a GAN which has been trained on a pre-trained GAN through transfer learning. The results obtained by Wang et al. [10] show a promising and powerful GAN that is able to generate high-quality images.

Additionally, implementing meta pseudo labels as done in the report by Pham et al. [6] might improve the performance. The main difference of using meta pseudo labels is that the teacher model receives feedback from the student's performance on the labeled dataset. The evaluation and training of the student model will therefore be conducted in a parallel fashion, further optimizing training. The generated pseudo labels will possibly be better and result in more efficient training for the student.

Lastly, incorporating saliency into our model might improve the prediction. Ghaeini et al. explain that saliency helps us to observe where the model is drawing evidence from to support its prediction [5]. In essence, incorporating this method would be equivalent to constraining the saliency and it would pay more attention to more decisive pieces of evidence. These models are commonly known as saliency attentive models and combined with ResNets results in SAM-ResNets.

7 Conclusion

The results show that a transfer learning implementation using ResNet yields high accuracy on the multi-class classification task. Even though it reaches a high accuracy, other model design choices should be evaluated for possible improvements.

Additionally, the report has shown that pseudo-labeling seems to be an efficient approach to semi-supervised learning, providing an alternative for training on less labeled data. However, more experiments and evaluations are needed to improve the labeling accuracy of the teacher. It has also been concluded that there most likely is a sweet spot for the amount of data available for a given task, for which pseudo-labeling is most effective.

References

[1] —. *Fine-Grained Image Classification on Oxford-IIIT Pet Dataset*. 2023. URL: `https://paperswithcode.com/sota/fine-grained-image-classification-on-oxford-1`.

[2] —. *FINETUNING TORCHVISION MODELS*. 2017. URL: https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html.

[3] Md Zahangir Alom et al. *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*. 2018. arXiv: 1803.01164 [cs.CV].

[4] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. *When Vision Transformers Outperform ResNets without Pre-training or Strong Data Augmentations*. 2022. arXiv: 2106.01548 [cs.CV].

[5] Reza Ghaeini et al. *Saliency Learning: Teaching the Model Where to Pay Attention*. 2019. arXiv: 1902.08649 [cs.CL].

[6] Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, Quoc V. Le. *Meta Pseudo Labels*. 2020. URL: https://arxiv.org/abs/2003.10580.

[7] jph00. *Computer vision intro*. URL: https://docs.fast.ai/tutorial.vision.html.

[8] Omkar M Parkhi et al. *The Oxford-IIIT Pet Dataset*. URL: https://www.robots.ox.ac.uk/~vgg/data/pets/.

[9] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, Quoc V. Le. *Self-training with Noisy Student improves ImageNet classification*. 2019. URL: https://arxiv.org/abs/1911.04252.

[10] Yaxing Wang et al. *Transferring GANs: generating images from limited data*. 2018. arXiv: 1805.01677 [cs.CV].

[11] Lilian Weng. *Learning with not Enough Data Part 1: Semi-Supervised Learning*. 2021. URL: https://lilianweng.github.io/posts/2021-12-05-semi-supervised/.