Transfer Learning

Team members: Theo Chen, Brian Guo, Xenia Konti

Abstract

In this project, we study **Transfer Learning**, a method of taking advantage of knowledge already gained in order to facilitate the training process of a new task. We provide a literature review of papers from various Machine Learning areas that have used Transfer Learning as a way of producing better final results. Then, we focus on two papers, "Transfer Learning via Learning to Transfer" and "SpotTune: Transfer Learning through Adaptive Fine-tuning", whose main goal is to learn what information is useful to be transferred and in what way can this transfer occur. We implement these ideas at a high level which allows us to provide some results and conclusions about them.

1 Introduction

Transfer Learning is a research area in Machine Learning that aims at using knowledge obtained during training over one problem and applying it to a different but similar one. It can be defined in terms of domains and tasks. A domain consists of the feature space, which follows a specific distribution, the label space, and a predictive function that connects the feature and label spaces. The task is to find the predictive function that satisfies the observed dataset. Given a source and a target domain and learning task, the goal of transfer learning is to help improve the learning of the target task by using information gained during the training of the source task.

1.1 Motivation and importance of the problem

In many cases, researchers have to learn tasks while the resources provided are inadequate for the training process. In other words, small sizes of datasets constitute an important obstacle to the proper training of a model. Furthermore, there are many cases in which the limited available time can result in a badly trained model. Both of these problems provide good motivation to researchers to keep studying transfer learning as a method of surpassing these difficulties.

Through transfer Learning, people can take advantage of knowledge already gained in order to accelerate the training time of a model and decrease the amount of data needed for the model to converge to a final state. It is important, however, that further research is conducted in this field in order to make sure that transferring knowledge from one problem to another will actually help the models and not lead to worse results.

2 Related works

2.1 Rich feature hierarchies for accurate object detection and semantic segmentation

The main purpose of this paper is to propose an algorithm that does semantic segmentation (recognizes the main features of an image), and object detection (classifies the features into specific classes). The proposed algorithm contains three different modules:

- The first module is a category-independent region generator that defines a set of segments in an image that need to be identified.

- The second module is a CNN that is used for feature extraction. More specifically, for each of the above-found regions, a 227x227 image version of the region is given as input to a CNN, which then generates as output a 4096-dimensional feature vector.
- The third and final module is a set of class-specific linear SVMs that is responsible for classifying every region. In terms of model architecture, this module is represented as an SVM weight matrix with 4096xN dimensions, where N is the number of classes.

As far as **transfer learning** is concerned, this method was applied for the initialization of the CNN's weights. In detail, the CNN was pre-trained on a labeled image dataset (supervised pre-training) with the target being to correctly classify the images. The pre-trained CNN and the one used on the target task (the paper's CNN) have the same architecture apart from the last layer. So, the authors of the paper used the first pre-trained layers of the CNN and then replaced the final one with a fully connected (N+1)-way classification layer that represents the SVM linear layer. Like it was mentioned before, N is the number of the problem's classes and the 'plus 1' is used for the regions that just show the background of the image and not a specific item in it.

The proposed algorithm has two main contributions. The first one is the unique architecture of the model, which allows the model to do both semantic segmentation and object detection. The second one is the fact that it tries to deal with scarce training data by pre-training an important module of the model. They measure their algorithm's performance by computing the mean average precision (mAP) and by showing that it outperforms other popular models on similar tasks. [3]

2.2 SpotTune: Transfer Learning through Adaptive Fine-tuning

A common problem arises when implementing practices of transfer learning. Given so many parameters in our pre-trained model, fine-tuning the whole network of parameters may often lead to overfitting. Many previous papers have proposed to freeze the last few layers of the network but even then choosing the right parameters to freeze in the first initial layers is manual and hard to optimize.

SpotTune approaches this problem with an input-dependent layer freezing technique. SpotTune does this by creating a policy network that determines which layers to freeze or fine-tune. The decision is determined by a lightweight neural network. This decision network is trained with the model, but due to the fact that these decision functions are non-differentiable, SpotTune uses Gumbel Softmax sampling.

The equation for the output becomes:

$$x_l = I_l(x)F_{tl}(x_{l-1}) + (1 - I_l(x))F_{fl}(x_{l-1}) + x_{l-1}$$

Where $I_l(x)$ is a binary random variable obtained from our policy network, determining whether we should freeze or fine-tune our layer. F_{tl} is our fine-tuned layer and F_{fl} is our frozen layer.

The SpotTune's main contribution is through creating a learnable optimizer to generate and decide on the best parameters to finetune in our transfer learning model. [5]

2.3 Transfer Learning via Learning to Transfer

L2T applies the educational psychology belief that humans decide what to transfer and learn using meta-cognitive reflection. This type of transfer learning that appears in human brains finds similar or adjacent knowledge within the source domain and the target domain to reduce the amount of target data needed. From the source domain, we can determine what knowledge has major overlaps with parts in the target domain.

Many researchers have tried to implement this idea, but achieving optimal performance has proved to be very inefficient through brute force. To counteract this, L2T proposes to learn from previous transfer learning experiences to determine which knowledge to transfer from the source domain to the target domain. More specifically, the L2T framework introduces two main ideas.

First, it suggests learning a function, called the reflection function, which gets as input the characteristics of a source task, target task and the information to be shared between them, and returns as output a value that represents the performance improvement of the algorithm when transfer learning occurs. The observed improvement is closely related to two factors:

- The difference between source and target domain when compared in a common latent space. In this paper, they propose mapping the two domains into the reproducing kernel Hilbert space and the compute the maximum mean discrepancy (MMD) between them.
- The discriminative ability of a target domain in the latent space. Specifically, the discriminant criterion of the target domain contains 2 components (1. Similar examples are close in the latent space 2. Different examples are far away from each other in the latent space)

The above aspects are combined in an optimization formulation, which attempts to minimize the MMD of the domains and maximize the target domain's discriminative ability, in an effort of minimizing the regression loss of the reflection function.

The second step of L2T is to be able to infer what and how to transfer in a case of a new pair of source-target domains. In detail, the information to be transferred is parameterized with a latent feature factor matrix W that can be used to specify domain invariant feature factors that should be shared across domains. After the reflection function is trained, for every new pair, the algorithm chooses the W that maximizes the expected improvement. [10]

2.4 Learning Transferable Features with Deep Adaptation Networks

It is usually the pattern that feature transferability drops as we go deeper into neural network layers. Latent representations from lower layers tend to reflect general characteristics while those from higher layers are more shaped by specific datasets and tasks. Therefore, the discrepancies between each domain will cause negative impacts on the training results, which gives us the desire to reduce dataset bias and enhance the transferability in task-specific layers. One of the intuitions is to train domain-invariant models from data, which links different domains in an isomorphic latent feature space. However, recent studies have demonstrated another feasible approach: domain adaptation. This paper focuses on this approach and proposes a new Deep Adaptation Network (DAN) architecture. In particular, hidden representations from higher layers get embedded in a kernel Hilbert space where the mean embeddings get matched. The authors also propose an optimal multi-kernel selection method to find the best kernel for matching.

The approach they propose is a multi-kernel variant of MMD proposed by Gretton et al [4], which is designed to jointly maximize the two-sample test power and minimize Type II error. In the multi-kernel version, a variable k is used for kernel selection to obtain optimal performance. Built on top of this principle, DAN adapted AlexNet [7] architecture with its first three convolution layers frozen and the other two fine-tuned to bridge domain discrepancy. While training, the network is fine-tuned on the source-labeled examples with the distributions of the source and target matched under the hidden representations. This is realized by adding an adaptation regularizer to the risk term. Efficacy is shown compared to the previous methods. The contribution of this paper is that it realizes a layer-wise deep adaptation and eliminates kernel sensitivity by introducing a kernel selection mechanism.[8]

2.5 Unsupervised Domain Adaptation with Residual Transfer Networks

Domain adaptation is capable of transferring a learner to a different domain when labels of the target task are not available. This paper proposes a new approach to domain adaptation called Residual Transfer Network (RTN), which is inspired by residual neural network (ResNet) [6]. This approach assumes that the source and target classifier only differ by a small residual function. Hence, by inserting several layers into deep networks, they are able to learn the residual function with reference to the target classifier, bridging the source and target classifier when back propagation runs. The authors claim that this approach generalizes to most feed-forward networks and the result is promising.

This approach generalizes better because it bypasses the assumption that the learned domain-invariant feature representations can be transferred directly to the target domain. This assumption introduces plenty of limitations when a data adaptation is attempted. Instead, it simultaneously learns from both domains and embeds the adaptations of both classifiers and transferable features in a unified deep structure. The target classifier is then tailored to target data with the learned low-density separation criteria. After the network is trained, the features of the inserted layers are fused and embedded into reproducing kernel Hilbert spaces where the distributions are matched.

The difference between the proposed approach and ResNet is that ResNet assumes training data and test data are drawn from identical distributions while the proposed approach recognizes bias and aims at adapting the network to datasets sampled from different distributions. It is also worth noting that although some previous studies (such as [2]) also adapt source classifiers to the target domain by adding a small perturbation function, they require labeled target data to train the models, while the proposed approach can be applied to unsupervised scenarios using a somehow more complicated way of bridging the classifiers.[9]

2.6 Provably Efficient Multi-Task Reinforcement Learning with Model Transfer

Transfer Learning, as a method of dealing with scarce training datasets, is widely used in many fields of machine learning and not just Deep Learning. Moreover, it is often used as a method of training concurrent models and not just transferring knowledge from one well-trained task to another task for which there aren't enough resources to train it. In this paper, they consider a Multi-task RL formulation where each task is learned by a different player. The main goal of the paper is to find a way to transfer learning between the players so that they can all learn their tasks faster.

The algorithm that is used for each individual task is called Strong-Euler. Basically, the player responsible for the task computes estimates of the rewards that certain actions can provide in specific states. At each time step, based on the previously computed estimates, the player chooses the action with the best expected reward for the current state and uses the observed reward to update the estimates. The difference in the algorithm that is proposed in this paper is that apart from the individual estimates, all the players compute global estimates based on the data selected by all. In other words, in this case, transfer learning happens in the form of actual data points.

The main contribution of this paper is that they provide theoretical proofs that show that transferring knowledge between players cannot make the algorithm work worse than the individual Strong-Euler that uses no transferring. Moreover, they show that in certain circumstances (in cases when there are a lot of common sub-optimal actions for all tasks) transferring learning can improve the algorithm quite a lot. In conclusion, this paper provides a theoretical analysis of transfer learning in order to guarantee that in this specific problem-formulation sharing of knowledge can only help the models converge faster and not worsen the final results.[11]

3 Details of the project

For this project, we implemented SpotTune for a time series problem. The model we designed was an LSTM with three layers of LSTM followed by two fully connected layers. For all the training processes, we used the Negative Log Likelihood Loss and the Adam optimizer to update the models' parameters.

In order to do transfer learning, we first had to train an LSTM model on an initial large data set, which we chose to be the UCI Daily and Sports Activities Data Set[1]. Then, we performed transfer learning with SpotTune from the pre-trained model to the new model whose goal is to perform classification on the UCI human activity dataset[1]. We compared these results to a benchmark of a model that was trained without transfer learning but with random initialization of its parameters.

In order to make transfer learning easier, we designed the two models (pre-trained and new) with the same architecture. For that reason, we had to process the two data sets, so that both of them had the same characteristics. In other words, both of them were left with 6 features and 125 time steps per data point (so the input size of the LSTMs was 6). Moreover, since the HAR dataset contained only 6 classes, in order for the LSTMs to have the same output size, we kept 6 out of the 19 classes of our source data set. These were the classes of lying on the back, standing in an elevator still, walking in a parking lot, exercising on a stepper, exercising on a cross trainer, and cycling on an exercise bike in a horizontal position. With this processed source data set we trained an initial model that was used as an initialization for our SpotTune model later on.

For the SpotTune algorithm, we had to design an agent that based on each input would choose which one of the frozen or tuned compartments of the model would be utilized, and also define what these compartments would be in our architecture. So first, we created an agent model with two layers of LSTM followed by two fully connected linear layers. Furthermore, we chose that the layers of our model that could have a frozen copy of them should be the LSTM ones.

In order to make the transfer learning more realistic, we kept only 50% of the target data set. Then we used 80% of it to train the SpotTune model and 20% for validation of the model per epoch. We added Gumbel softmax on the agent model's output to obtain the policy that would decide which of the frozen/tuned layers of the model would be used to compute the output of each input of our Human Activity Recognition data set.

3.1 Contribution of each member of the team

- Each team member chose two research papers about transfer learning methods to review
- Every team member worked together to code the module
- All team members compiled the final paper together

4 Experimental results

The first thing that we observed was that SpotTune is quite a computationally heavy model. Since it trains a classifier and an agent model simultaneously, and since for every input it changes the structure of the classifier based on the proposed agent's policy, it is understandable that every epoch needs a lot of time to train and backpropagate the loss to the involved layers. In order to have some initial results and thoughts about the algorithm, we trained the model only for ten epochs. Moreover, in order for the comparison to the benchmark to be fair, we trained the benchmark for ten epochs as well.

From the results of the training process (that can be observed in the two figures as well), it is evident that both of the two models did not reach a level of overfitting and they could be trained for more epochs. Setting this aside, it was also evident that the benchmark performed better than SpotTune. Even for the metrics on the testing set, we have the following:

- $\frac{\text{Test Loss on Spottune}}{\text{Test Loss on Benchmark}} = \frac{1.4175454378128052}{1.146884560585022} = 1.23$
- $\frac{\text{Test Accuracy on Spottune}}{\text{Test Accuracy on Benchmark}} = \frac{0.31964709874448594}{0.34781133355955207} = 0.91$

We had two main explanations for why this happened:

1. The agent of the SpotTune algorithm was initialized randomly. So, in the beginning, it would not necessarily make the correct decisions and that would affect both the result for the given time step and the training of the classifier for the future time steps.
2. It is not easy to implement Spottune on an LSTM model. Each layer of LSTM contains both long and short-term memory information. Freezing a whole layer affects quite a large and important component of the model.
3. While Spottune aims at preserving latent representations that reflect shared features and fine-tuning representations that are specific to the source dataset, long and short-term memory mechanism also embodies similar philosophy in that long-term memory memorizes general characteristics and short-term memory is susceptible to changes in data. We conjure that these two approaches might not be orthogonal and could lead to intervention when they are applied simultaneously.

5 Concluding remarks

In this project, we explored a broad set of topics in **Transfer Learning** including R-CNN, SpotTune, L2T, Data adaptation through DAN and RTN, and transferred reinforcement learning models. We delved deep into the principle underneath SpotTune and implemented the method from scratch. We successfully obtained results from SpotTune applied on a three-layer LSTM network and conducted an evaluation. Concerning the fact that the SpotTune implementation did not show superiority over the default benchmark, we provided our interpretations and analysis. Three potential reasons are discussed as explanations. Our project successfully demonstrated key takes in the field of **Transfer Learning** and examined the feasibility of SpotTune under our specifications. Readers can take our results and evaluations as a reference.

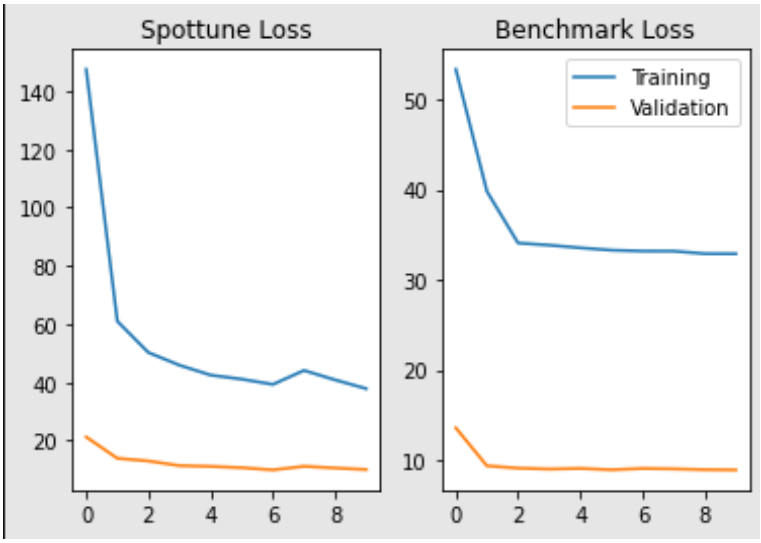

Figure 1: Training and Validation Loss for both Spottune and Benchmark models

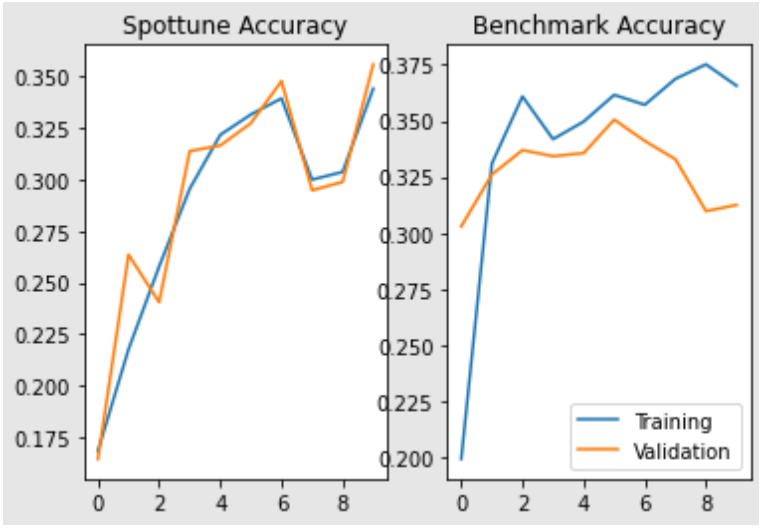

Figure 2: Training and Validation Accuracy for both Spottune and Benchmark models

References

[1] Dheeru Dua and Casey Graff. Uci machine learning repository, 2017.

[2] Lixin Duan, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. Domain adaptation from multiple sources via auxiliary classifiers. *ICML*, 2009.

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[4] A. Gretton, B. Sriperumbudur, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, and K Fukumizu. Optimal kernel choice for large-scale two-sample tests. *NIPS*, 2012.

[5] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogério Schmidt Feris. Spottune: Transfer learning through adaptive fine-tuning. *CoRR*, abs/1811.08737, 2018.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[8] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *ICML*, 2015.

[9] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Unsupervised domain adaptation with residual transfer networks. *NIPS*, 2016.

[10] Ying Wei, Yu Zhang, and Qiang Yang. Learning to transfer. *CoRR*, abs/1708.05629, 2017.

[11] Chicheng Zhang and Zhi Wang. Provably efficient multi-task reinforcement learning with model transfer. *Advances in Neural Information Processing Systems*, 34:19771–19783, 2021.