What is Transfer Learning?

Anton Zhitomirsky

January 4, 2024

TODO: [6, 17, 11, 4, 9, 8, 1, 13, 10, 14, 12]

1 What is Transfer Learning?

We can use models that have been trained for one task, and use it as a starting point for a model on a second task. This can be useful when the second task is similar to the first task, or when there is a limited amount of data available [16]. The intuitive reason why transfer learning works is because the early layers of the network a deep learning model attempts to learn very low-level features. All these features occur regardless of the exact cost function or image dataset [16], which makes it ideal for transfer learning because the set of low-level features learnt will be similar.

This task can be viewed as a regularization task; 'The assumption of similarity between the tasks represents a more sophisticated form of inductive bias compared to simple regularization, and this explains the improved performance resulting from the use of the additional data' [2]. We can use the assumption of similarity especially when the distribution changes. Instead of rebuilding statistical models from scratch using newly collected training data (this may be impossible or expensive) we can transfer knowledge across domains [12].

It has been shown to work in medical contexts as well in [7] of 332 abdominal liver CT scans and even outperformed traditional frameworks[1]. In this paper they compared the performance of networks with (1) training from scratch (random initialization), (2) fine tuning the ImageNet weights, (3) fine tuning weights of a self-trained lesion segmentation model. "Using transfer learning generally improved weight initialization and resulted in faster convergence providing stronger and more robust representation" [7].

1.1 Formally Defining the Problem

As inspired by [6] and [12].
Definition 1: A **domain**

$$\mathcal{D} = \{\mathcal{X}, P(X)\} \tag{1}$$

With feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ where X is defined as an instance set, and $X = \{x_1, x_2, \cdots, x_n\} \in \mathcal{X}$.

Definition 2: A **task**

[1] `https://arxiv.org/pdf/2003.13912.pdf`

$$\mathcal{T} = \{\mathcal{Y}, f(\cdot)\} \tag{2}$$

composed of a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$. Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ the sample data consists of pairs $\{x_i, y_i\}$ where $x_i \in X$ and $y_i \in \mathcal{Y}$. The objective function f is supposed to learn from sample data to predict the corresponding label for the new instances. f can be rewritten as $f(x) = P(y|x)$.

Definition 3: Given a domain $\mathcal{D}_S$ with corresponding source tasks $\mathcal{T}_S$, and a target domain $\mathcal{D}_T$ with corresponding target task $\mathcal{T}_T$, **transfer learning** aims to transfer the related knowledge to boost the performance of the target predictive function

$$f_T(\cdot) \tag{3}$$

1.2 Advantages

- By using the learned features from the first task as a starting point, the model can learn more quickly and effectively on the second task [16] (it already has a good understanding of the features and patterns in the data).
- can also help to prevent overfitting, as the model will have already learned general features that are likely to be useful in the second task [16].
- When we look at convolutional networks we will see that many image processing tasks require similar low-level features corresponding to the early layers of a deep neural network, whereas later layers are more specialized to a particular task, making such networks well suited to transfer learning applications. [2]
- Three measures can be analysed for improvements [15];
  - The initial performance achievable in the target task using only the transferred knowledge, before any further learning is done, compared to the initial performance of an ignorant agent.
  - The amount of time it takes to fully learn the target task given the transferred knowledge compared to the amount of time to learn it from scratc
  - The final performance level achievable in the target task compared to the final level without transfer

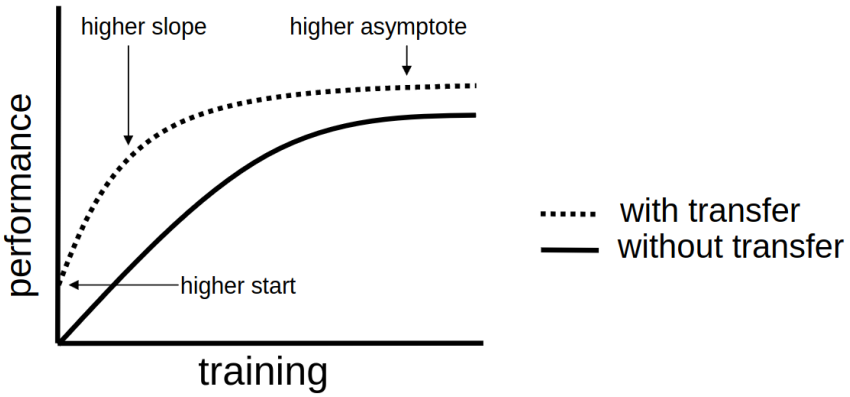


Figure 1: Figure taken from [15] showing three ways in which transfer might improve learning

1.3 Disadvantages

- Domain mismatch if the two models are trained on domains that are vastly different or with a different data distribution [16].
- Overfitting 'Transfer learning can lead to overfitting if the model is fine-tuned too much on the second task, as it may learn task-specific features that do not generalize well to new data.' [16].

1.4 General Approach

This boils down to three questions: (1) What to transfer, (2) How to transfer, and (3) When to transfer.
Some knowledge may be common between different domains such that they may help improve performance for the target domain or task. When the source domain and target domain are not related to each other, brute-force transfer may be unsuccessful. In the worst case, it may even hurt the performance of learning in the target domain, a situation which is often referred to as negative transfer (see Section 2).

1.4.1 Strategy

1. Obtain a pre-trained 'base' model. This model has been trained on extensive data and has identified general features and patterns relevant to numerous related jobs.
2. Identify the transfer-layers. These capture generic information relevant to the new task as well as the previous one. These are 'frozen' in the final deliverable (see Figure 2) as we want to preserve the low-level learnt features. A layer is frozen or 'fixed' when it is no longer avaialble for training, hence the weights of these layers will not be avaialble for updates.
3. Fine-tune and retrain the remaining layers. The goal is to preserve the knowledge from the pre-training while enabling the model to modify its parameters to better suit the demands of the current assignment. We can either
    - freeze a few layers of the pre-trained model and train other layers on our new dataset for the new task

- or make a new model, but also take out some features from the layers in the pre-trained model and use them in a newly created model

The number of frozen layers depend on how much you want to inherit from the pre-trained model. If the networks are significantly different (i.e. use a human face detector to detect cars) selecting many layers for freezing will not only give low level features but also give high-level features like nose, eyes, etc which are useless for new dataset (car detection). Therefore, in this case you only copy the low level features [16].

```
# freezing layers in python
for layer in base_model.layers:
    layer.trainable = False
```

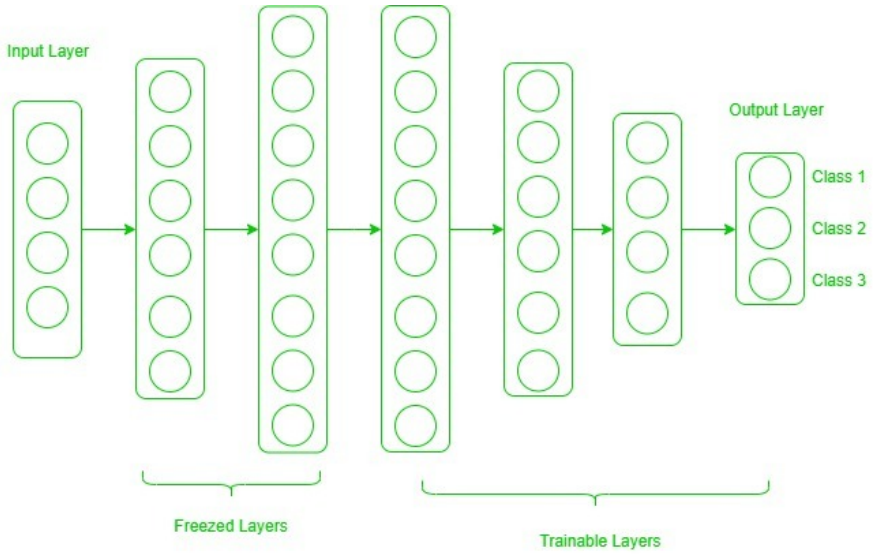


Figure 2: Figure taken from [16] illustrating frozen and trainable layers from pre-trained and fine-tuned network.

1.5 Recommendations

- **The target dataset is small and similar to the base network dataset**: There may be an issue of overfitting because fine-tune the pre-trained network with the target dataset may not generalise to the global population. Also, there may be some changes in the number of classes in the target task; remove the fully connected layers from the end and add a new fully connected layer satisfying the number of new classes. Now, we freeze the rest of the model and only train newly added layers [16].

- **The target dataset is small and different from the base network dataset**: Since the target dataset is different, using high-level features of the pre-trained model will not be useful. In such a case, remove most of the layers from the end in a pre-trained model, and add new layers a satisfying number of classes in a new dataset. This way we can use low-level features from the pre-trained model and train the rest of the layers to fit a new dataset. Sometimes, it is beneficial to train the entire network after adding a new layer at the end [16].

- **Set the learning rate to be low**: It is essential to utilize a lower learning rate when compiling the new model because you are training a much larger model and want to readjust the pretrained weights. If not, your model may rapidly become overfit [16]. Furthermore, this is done for a low number of iterations for the same reasons [2].
- **Don't use stochastic gradient descent algorithms to the whole network**; it is much more efficient to send the new training data once through the fixed pre-trained network so as to evaluate the training inputs in the new representation. Iterative gradient-based optimization can then be applied just to the smaller network consisting of the final layers.

2 Avoiding Negative Transfer

If a transfer method actually decreases performance, then negative transfer has occurred. One of the major challenges in developing transfer methods is to produce positive transfer between appropriately related tasks while avoiding negative transfer between tasks that are less related [15].

2.1 Fixes

Reference [15] contains a lot of useful sources regarding sources of different ideas surrounding transfer learning in different domains that might be extensible to a medical domain. This source was found through the source [3].

2.1.1 Rejecting Bad Information

The goal in this approach is to minimize the impact of bad information, so that the transfer performance is at least no worse than learning the target task without transfer. At the extreme end, an agent might disregard the transferred knowledge completely, but some methods also allow it to selectively reject parts and keep other parts [15].

2.1.2 Choosing a subtask

Eaton and DesJardins "propose choosing from among candidate solutions to a source task rather than from among candidate source tasks. Their setting is multi-resolution learning, where a classification task is solved by an ensemble of models that vary in complexity. Low-resolution models are simple and coarse, while higher-resolution models are more complex and detailed" [15].

2.1.3 Modelling Task Similarity

Given multiple candidate source tasks, it may be beneficial to use several or all of them rather than to choose just one.
"Ruckert and Kramer look at inductive transfer via kernel methods. They learn a meta-kernel that serves as a similarity function between tasks. Given this and a set of kernels that perform well in source tasks, they perform numerical optimization to construct a kernel for a target task. This approach determines the inductive bias in the target task (the kernel) by combining information from several source tasks whose relationships to the target are known" [15].

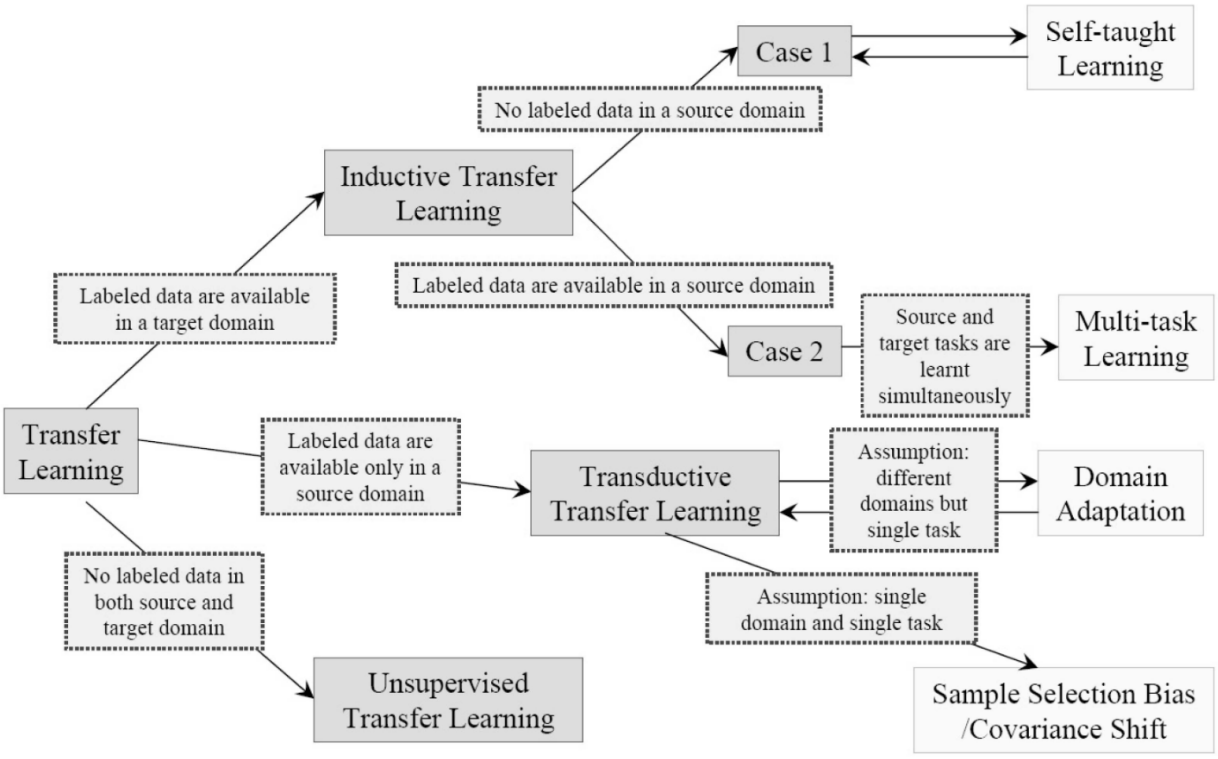
3 Transfer Learning Settings

Learning Settings		Source and Target Domains	Source and Target Tasks
Traditional Machine Learning		the same	the same
Transfer Learning	*Inductive Transfer Learning* / *Unsupervised Transfer Learning*	the same	different but related
		different but related	different but related
	Transductive Transfer Learning	different but related	the same

(a) Relationship between Traditional Machine Learning and Various Transfer Learning Settings

Transfer Learning Settings	Related Areas	Source Domain Labels	Target Domain Labels	Tasks
Inductive Transfer Learning	Multi-task Learning	Available	Available	Regression, Classification
	Self-taught Learning	Unavailable	Available	Regression, Classification
Transductive Transfer Learning	Domain Adaptation, Sample Selection Bias, Co-variate Shift	Available	Unavailable	Regression, Classification
Unsupervised Transfer Learning		Unavailable	Unavailable	Clustering, Dimensionality Reduction

(b) Different Settings of Transfer Learning



(c) An overview of different settings for transfer learning

Transfer Learning Approaches	Brief Description
Instance-transfer	To re-weight some labeled data in the source domain for use in the target domain [6], [28], [29], [30], [31], [24], [32], [33], [34], [35].
Feature-representation-transfer	Find a "good" feature representation that reduces difference between the source and the target domains and the error of classification and regression models [22], [36], [37], [38], [39], [8], [40], [41], [42], [43], [44].
Parameter-transfer	Discover shared parameters or priors between the source domain and target domain models, which can benefit for transfer learning [45], [46], [47], [48], [49].
Relational-knowledge-transfer	Build mapping of relational knowledge between the source domain and the target domains. Both domains are relational domains and i.i.d assumption is relaxed in each domain [50], [51], [52].

(d) Different Approaches to Transfer Learning

	Inductive Transfer Learning	Transductive Transfer Learning	Unsupervised Transfer Learning
Instance-transfer	√	√	
Feature-representation-transfer	√	√	√
Parameter-transfer	√		
Relational-knowledge-transfer	√		

(e) Different Approaches Used in Different Setings

Figure 3: Taken from [12]

3.1 Inductive Transfer Learning

Relevant for our case is the branch from Figure 3(c) is 'Labeled data are avaialble in a target domain'. Therefore we lookup these cases only.

3.2 Instance-transfer Approach

From Figure 3(e) we have a source domain data which cannot be reused directly, however, there are certain parts of the data that can still be reused together with a few labeled data in the target domain. Note that there exist boosting algorithms TrAdaBoost (see [5] which details the algorithm for updating weight vectors).

See further info at [12]

4 Sources of pre-trained models

From [3] we have sources of pre-trained networks available at Model-Zoo Github as well as a walk-through.

References

[1] Nikolas Adaloglou. *Transfer learning in medical imaging: classification and segmentation.* 2020. URL: `https://theaisummer.com/medical-imaging-transfer-learning/`.

[2] Christopher M. Bishop and Hugh Bishop. *Deep Learning, Foundations and Concepts.* Springer, 2023.

[3] Jason Brownlee. "A Gentle Introduction to Transfer Learning for Deep Learning". In: (2019). URL: `https://machinelearningmastery.com/transfer-learning-for-deep-learning/`.

[4] Sasank Chilamkurthy. *Transfer Learning for Computer Vision Tutorial.* URL: `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`.

[5] W. Dai et al. *Boosting for Transfer Learning*. Tech. rep. 2007. URL: https://cse.hkust.edu.hk/~qyang/Docs/2007/tradaboost.pdf.

[6] Abolfazl Farahani et al. *A Concise Review of Transfer Learning*. Tech. rep. 2021. URL: https://arxiv.org/abs/2104.02144v1.

[7] Michal Heker and Hayit Greenspan. *Joint Liver Lesion Segmentation and Classification via Transfer Learning*. Tech. rep. 2020. URL: https://arxiv.org/pdf/2004.12352.pdf.

[8] Hee E. Kim et al. *Transfer learning for medical image classification: a literature review*. Tech. rep. 2022. URL: https://bmcmedimaging.biomedcentral.com/articles/10.1186/s12880-022-00793-7.

[9] Pedro Marcelino. *Transfer learning from pre-trained models*. 2018. URL: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751.

[10] Basil Mustafa et al. *Supervised Transfer Learning at Scale for Medical Imaging*. 2021. URL: https://arxiv.org/abs/2101.05913.

[11] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. *What is being transferred in Transfer Learning?* Tech. rep. 2020. URL: https://arxiv.org/abs/2008.11687.

[12] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: (2009). URL: https://ieeexplore.ieee.org/abstract/document/5288526.

[13] Maithra Raghu et al. *Transfusion: Understanding Transfer Learning for Medical Imaging*. 2019. URL: https://arxiv.org/abs/1902.07208.

[14] Abdel Aziz Taha and Allan Hanbury. "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool". In: (2015). URL: https://ncbi.nlm.nih.gov/pmc/articles/PMC4533825/.

[15] Lisa Torrey and Jude Shavlik. *Transfer Learning*. Tech. rep. University of Wisconsi, 2009. URL: https://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf.

[16] "What is Transfer Learning?" In: (). URL: https://www.geeksforgeeks.org/ml-introduction-to-transfer-learning/.

[17] Fuzhen Zhuang et al. *A Comprehensive Survey on Transfer Learning*. Tech. rep. 2019. URL: https://arxiv.org/abs/1911.02685.