Transfer Learning

Niklas Donges Follow
Apr 23, 2018 · 8 min read

Transfer Learning is the reuse of a pre-trained model on a new problem. It is currently very popular in the field of Deep Learning because it enables you to train Deep Neural Networks with comparatively little data. This is very useful since most real-world problems typically do not have millions of labeled data points to train such complex models. This blog post is intended to give you an overview of what Transfer Learning is, how it works, why you should use it and when you can use it. It will introduce you to the different approaches of Transfer Learning and provide you with some resources on already pre-trained models.

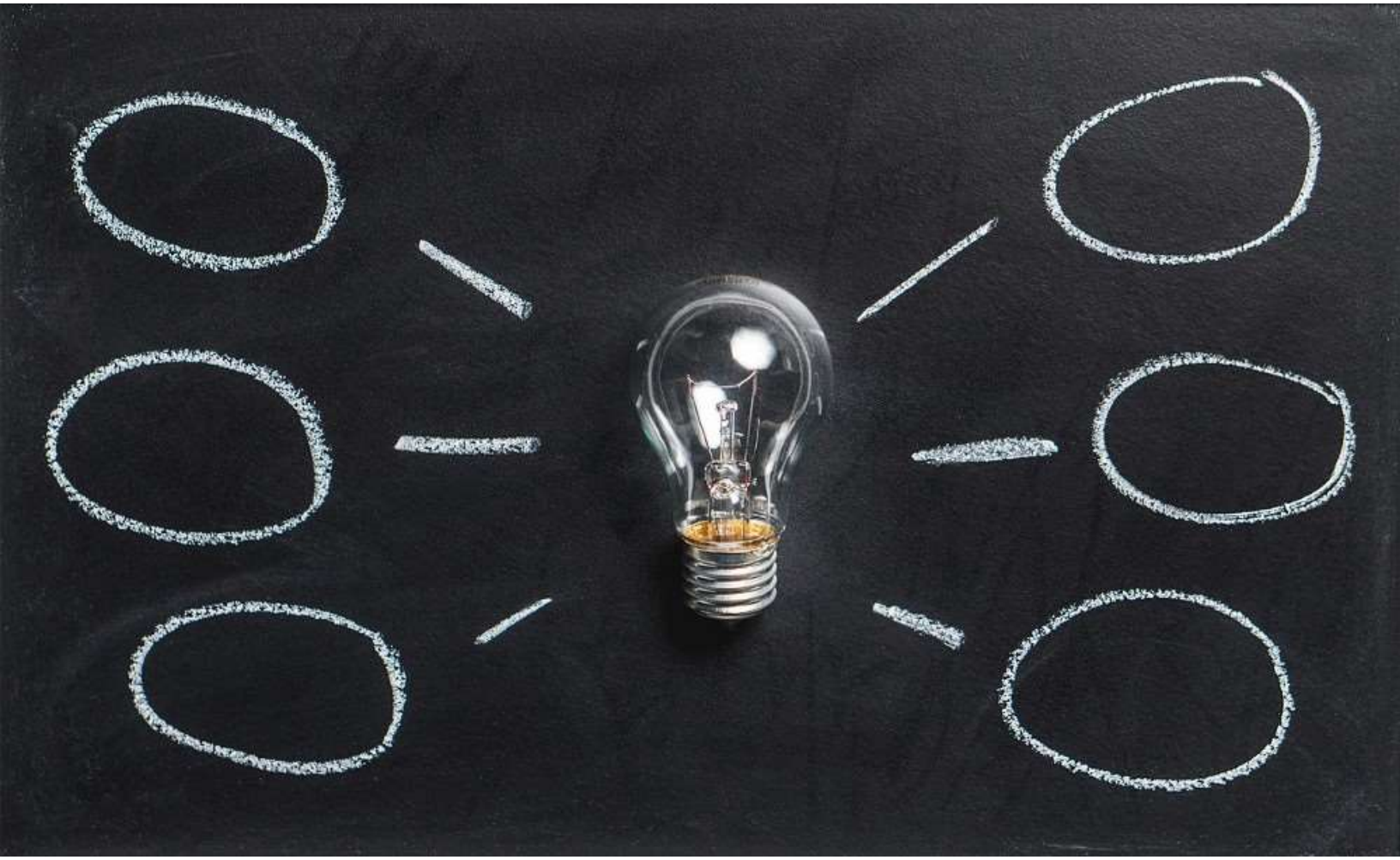


Table of Contents:

- **How it works**
- **Why it is used?**
- **When you should use it**
- **Approaches to Transfer Learning (Training a Model to Reuse it, Using a Pre-Trained Model, Feature Extraction)**
- **Summary**

What is it?

In Transfer Learning, the knowledge of an already trained Machine Learning model is applied to a different but related problem. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the knowledge that the model gained during its training to recognize other objects like sunglasses.

With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another. We transfer the weights that a Network has learned at Task A to a new Task B.

The general idea is to use knowledge, that a model has learned from a task where a lot of labeled training data is available, in a new task where we don't have a lot of data. Instead of starting the learning process from scratch, you start from patterns that have been learned from solving a related task.

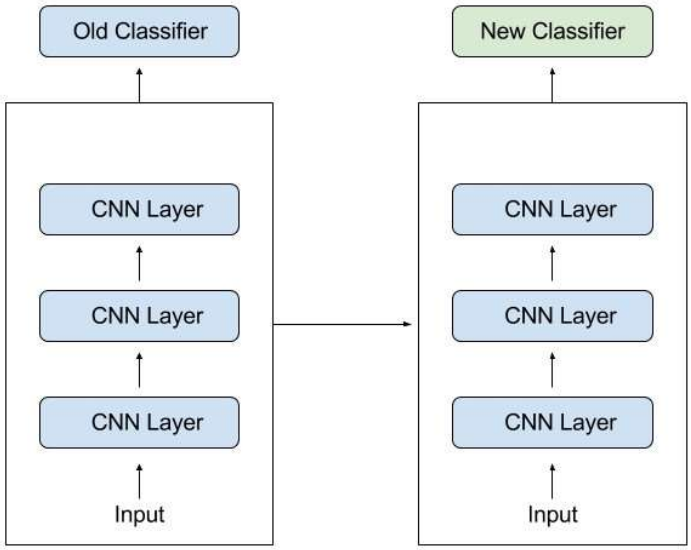
Transfer Learning is mostly used in Computer Vision and Natural Language Processing Tasks like Sentiment Analysis, because of the huge amount of computational power that is needed for them.

It is not really a Machine Learning technique. Transfer Learning can be seen as a 'design methodology' within Machine Learning like for example, active learning. It is also not an exclusive part or study-area of Machine Learning. Nevertheless, it has become quite popular in the combination with Neural Networks, since they require huge amounts of data and computational power.

How it works

For example, in computer vision, Neural Networks usually try to detect edges in their earlier layers, shapes in their middle layer and some task-specific features in the later layers. With transfer learning, you use the early and middle layers and only re-train the latter layers. It helps us to leverage the labeled data of the task it was initially trained on.

Let's go back to the example of a model trained for recognizing a backpack on an Image, which will be used to identify Sunglasses. In the earlier layers, the model has learned to recognize objects and because of that, we will only re-train the latter layers, so that it will learn what separates sunglasses from other objects.



In Transfer Learning, we try to transfer as much knowledge as possible from the previous task, the model was trained on, to the new task at hand. This knowledge can be in various forms depending on the problem and the data. For example, it could be how models are composed which would allow us to more easily identify novel objects.

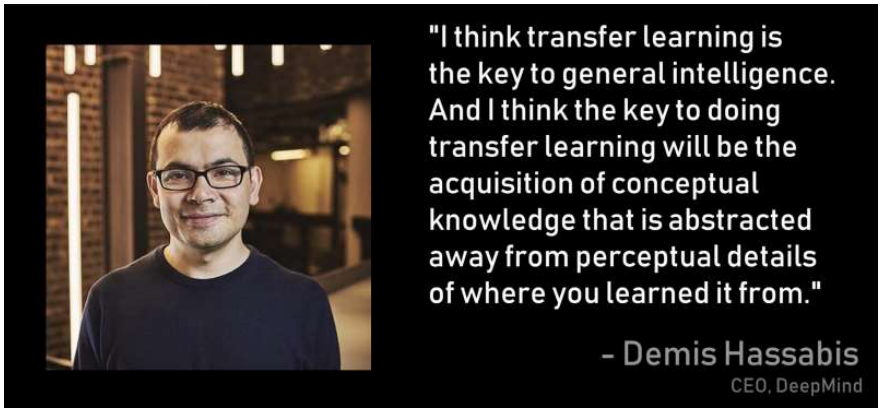
Why it is used?

Using Transfer Learning has several benefits that we will discuss in this section. The main advantages are basically that you save training time,

that your Neural Network performs better in most cases and that you don't need a lot of data.

Usually, you need a lot of data to train a Neural Network from scratch but you don't always have access to enough data. That is where Transfer Learning comes into play because with it you can build a solid machine Learning model with comparatively little training data because the model is already pre-trained. This is especially valuable in Natural Language Processing (NLP) because there is mostly expert knowledge required to created large labeled datasets. Therefore you also save a lot of training time, because it can sometimes take days or even weeks to train a deep Neural Network from scratch on a complex task.

According to Demis Hassabis, the CEO of DeepMind Technologies, Transfer is also one of the most promising techniques that could someday lead us to Artificial General Intelligence (AGI):



When you should use it

As it is always the case in Machine Learning, it is hard to form rules that are generally applicable. But I will provide you with some guidelines.

You would typically use Transfer Learning when (a) you don't have enough labeled training data to train your network from scratch and/or (b) there already exists a network that is pre-trained on a similar task, which is usually trained on massive amounts of data. Another case where its use would be appropriate is when Task-1 and Task-2 have the same input.

If the original model was trained using TensorFlow, you can simply restore it and re-train some layers for your task. Note that Transfer Learning only works if the features learned from the first task are general, meaning that they can be useful for another related task as well. Also, the input of the model needs to have the same size as it was

initially trained with. If you don't have that, you need to add a preprocessing step to resize your input to the needed size.

Approaches to Transfer Learning

Now we will discuss different approaches to Transfer Learning. Note that these have different names throughout literature but the overall concept is mostly the same.

1. Training a Model to Reuse it

Imagine you want to solve Task A but don't have enough data to train a Deep Neural Network. One way around this issue would be to find a related Task B, where you have an abundance of data. Then you could train a Deep Neural Network on Task B and use this model as starting point to solve your initial Task A. If you have to use the whole model or only a few layers of it, depends heavily on the problem you are trying to solve.

If you have the same input in both Tasks, you could maybe just reuse the model and make predictions for your new input. Alternatively, you could also just change and re-train different task-specific layers and the output layer.

2. Using a Pre-Trained Model

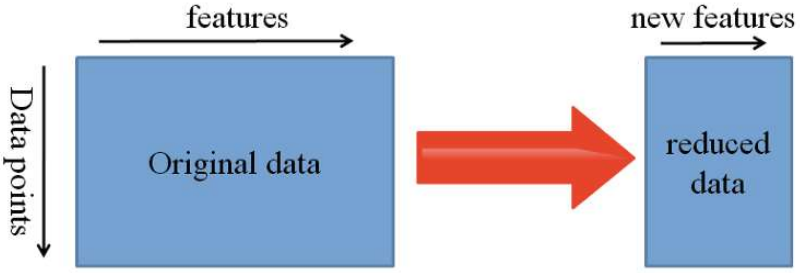
Approach 2 would be to use an already pre-trained model. There are a lot of these models out there, so you have to do a little bit of research. How many layers you reuse and how many you are training again, depends like I already said on your problem and it is therefore hard to form a general rule.

Keras, for example, provides nine pre-trained models that you can use for Transfer Learning, Prediction, feature extraction and fine-tuning. You can find these models and also some brief tutorial on how to use them here.

There are also many research institutions that released models they have trained. This type of Transfer Learning is most commonly used throughout Deep Learning.

3. Feature Extraction

Another approach is to use Deep Learning to discover the best representation of your problem, which means finding the most important features. This approach is also known as Representation Learning and can often result in a much better performance than can be obtained with hand-designed representation.



Most of the time in Machine Learning, features are manually hand-crafted by researchers and domain experts. Fortunately, Deep Learning can extract features automatically. Note that this does not mean that Feature Engineering and Domain knowledge isn't important anymore because you still have to decide which features you put into your Network. But Neural Networks have the ability to learn which features, you have put into it, are really important and which ones aren't. A representation learning algorithm can discover a good combination of features within a very short timeframe, even for complex tasks which would otherwise require a lot of human effort.

The learned representation can then be used for other problems as well. You simply use the first layers to spot the right representation of features but you don't use the output of the network because it is too task-specific. Simply feed data into your network and use one of the intermediate layers as the output layer. This layer can then be interpreted as a representation of the raw data.

This approach is mostly used in Computer Vision because it can reduce the size of your dataset, which decreases computation time and makes it more suitable for traditional algorithms as well.

Popular Pre-Trained Models

There are a some pre-trained Machine Learning models out there that became quite popular. One of them is the Inception-v3 model, which was trained for the ImageNet "Large Visual Recognition Challenge". In this challenge, participants had to classify images into 1000 classes, like "Zebra", "Dalmatian", and "Dishwasher".

Here you can see a very good tutorial from TensorFlow on how to retrain image classifiers.

Microsoft also offers some pre-trained models which are available for both R and Python development, through the MicrosoftML R package and the microsoftml Python package.

Other quite popular models are ResNet and AlexNet. I also encourage you to visit pretrained.ml which is a sortable and searchable

compilation of pre-trained deep learning models, along with demos and code.

Summary

In this post, you have learned what Transfer Learning is and why it matters. You also discovered how it is done along with some of its benefits. We talked about why it can reduce the size of your dataset, why it decreases training time and why you also need less data when you use it. We discussed when it is appropriate to do Transfer Learning and what are the different approaches to it. Lastly, I provided you with a collection of models that are already pre-trained.

Resources

https://medium.com/@14prakash/transfer-learning-using-keras-d804b2e04ef8

http://ruder.io/transfer-learning/

https://www.datacamp.com/community/tutorials/transfer-learning

https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems

Deep Learning (Adaptive Computation and Machine Learning)