# Transfer Learning

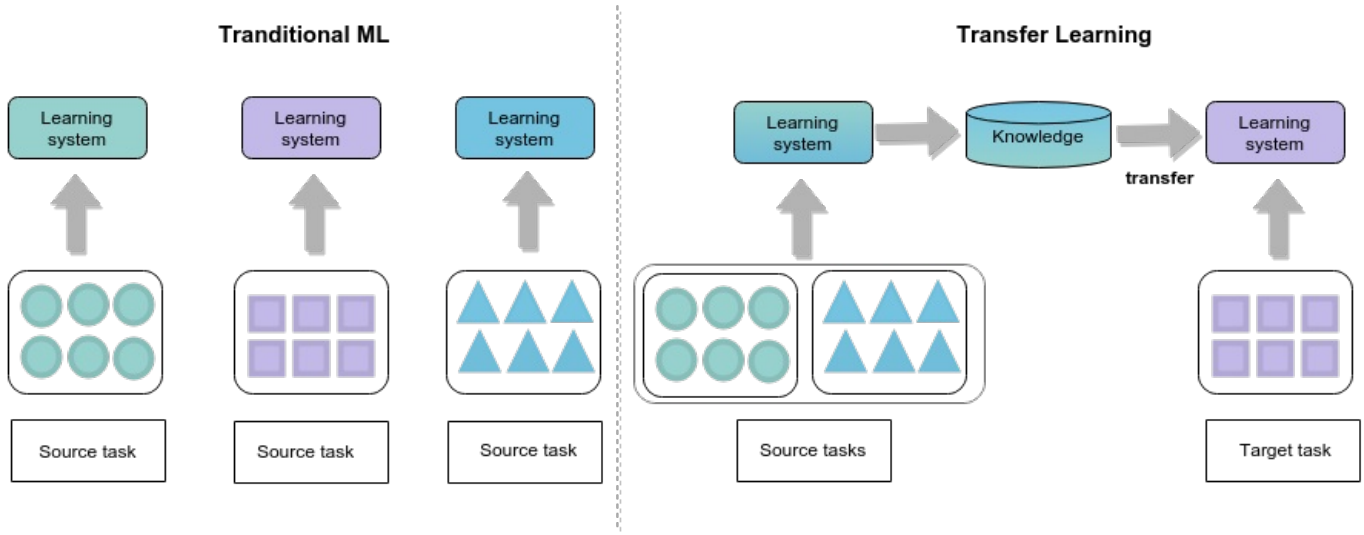
An introduction to transfer learning, its importance, and applications.

We'll cover the following



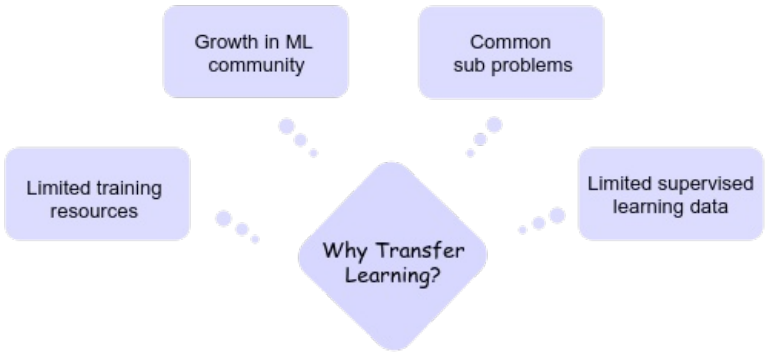
## What is transfer learning? #

Transfer learning is the task of using a pre-trained model and applying it to a new task, i.e., transferring the knowledge learned from one task to another. This is useful because the model doesn't have to learn from scratch and can achieve higher accuracy in less time as compared to models that don't use transfer learning.




### Motivation #

The use of transfer learning in the machine learning domain has surged in the last few years. The following are the top reasons:




Motivation for transfer learning

1. **Growth in the ML community and knowledge sharing**: The research and investments by top universities and tech companies have grown exponentially in the last few years and there is also a strong desire to share state-of-the-art models and datasets with the community. This allows people to utilize pre-trained models in a specific area bootstrap quickly.

2. **Common sub-problems**: Another key motivator is that many problems share common sub-problems, e.g., in all visual understanding and prediction areas, tasks such as finding edges, boundaries, and background are common sub-problems. Similarly, in the text domain, the semantic understanding of textual terms can be helpful in almost all problems where the user is represented by text terms, including search, recommendation systems, ads, etc.

3. **Limited supervised learning data and training resources**: Many real-world applications are still mapped onto supervised learning problems where the model is asked to predict a label. One key problem is the limited amount of training data available for models to generalize well. One key advantage of doing transfer learning is that we have the ability to start learning from pre-trained models, and hence, we can utilize the knowledge from similar domains.

   Self-supervised learning models are able to utilize massive available datasets for text and image representation, e.g., Word2vec embedding models don't need any manual labels and can use the books and Wikipedia data to build a semantic understanding of terms effectively. Once we train a model for a certain representation, it can be utilized and help in many other supervised learning tasks.

   Transfer learning also optimizes training resources, and it helps teams that don't have massive computing resources available. For instance, Google can train a BERT model on billions of examples with its massive computing power, but others are going to find it challenging to train similar optimized models. With transfer learning, we don't have to reinvest those resources and can just plug in the output of the BERT model or use it as a sub-model in our training process. We discussed this concept in detail earlier in our embedding (https://www.educative.io/collection/page/10370001/6237869033127936/6130870193750016) lesson discussion.

# Techniques for transfer learning utilization #

The transfer learning technique can be utilized in the following ways:

- **Extract features from useful layers**

  Keep the initial layers of the pre-trained model and remove the final layers. Add the new layer to the remaining chunk and train them for final classification.

- **Fine-tuning**

  Change or tune the existing parameters in a pre-trained network, i.e., optimizing the model parameters during training for the supervised prediction task. A key question with fine-tuning the model is to see how many layers can we freeze and how many final layers we want to fine-tune. This requires understanding the network structure of the model and role of each layer, e.g., for the image classification model we used in the Image data example, once we understand the convolution, pooling, and fully connected layers, we can decide how many final layers we need to fine-tune for our model training process.

Transfer learning technique can be utilized in one or both of the above ways depending on the following two factors:

1. **Size of our supervised training dataset**
   How much labeled data do we possess to optimize the model? Do we have 100k examples, 1 million examples, 10 million examples? This is an important question for deciding on the approach that we want to use in utilizing the pre-trained model.

   *Training data is limited:* In case of a limited amount of specialized training data, we can either go with the approach of freezing all the layers and using the pre-trained model for feature generation or fine-tuning only the final layers.

   *Training data is plenty:* If we have a significant amount of training data (e.g. one million+ examples), we have the choice to play around with multiple ideas. We can start with just freezing the model, fine-tuning only final layers, or we can retrain the whole model to adjust weights for our specialized task.

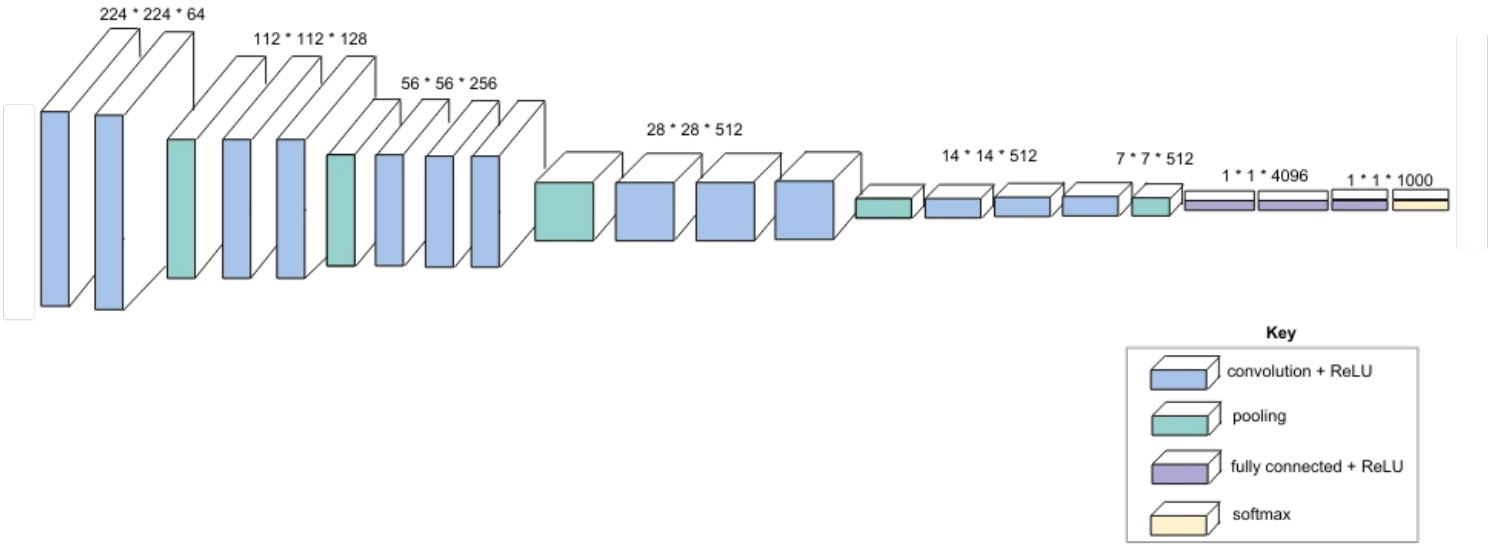
2. **Similarity of prediction tasks**

   The similarity of learning tasks can also guide us on whether we can simply use the model as it is or need to fine-tune the model for our new prediction task. For example, if we built a classifier for cars and now we want to use it for trucks, there is a good chance that many of the features are going to be common and we don't have to fine-tune much. Here, we can utilize the pre-trained model as it is and build our models on top of it (i.e., utilizing the output of pre-trained models as features).

# Applications #

## Computer vision problems #

Let's go over an example problem where we are trying to build a classifier for medical imaging data and we have 100k manual labelled examples for training our model. Given significant amount of research done in ImageNet data classifier, we can pick one pre-trained ImageNet classifier and start building on top of it.



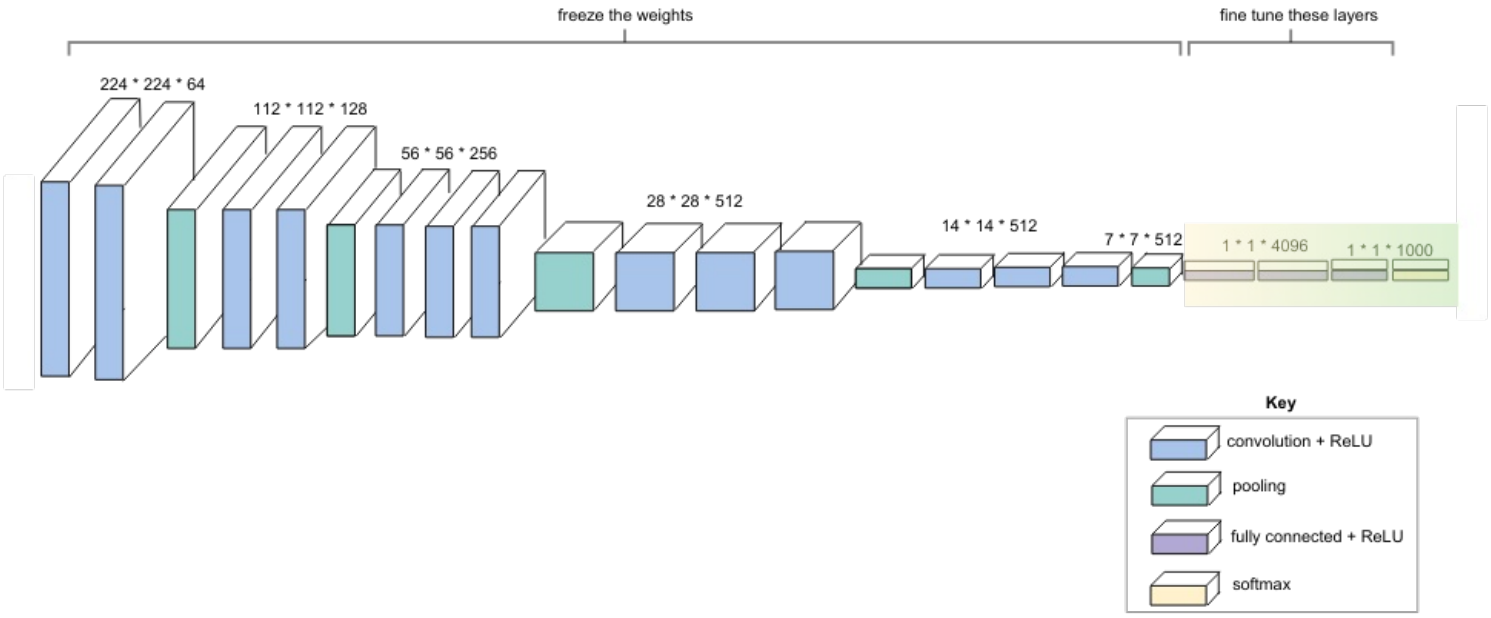

VGG16 architecture

The convolutional filters in a trained convolutional neural network (CNN) are arranged in a kind of hierarchy. The filters in the first layer often detect edges or blocks of color. The second layer's filters can detect features like shapes. All of them are very general features that are useful in analyzing any image in any dataset. The filters in the last layers are more specific. Let's go over all of the freezing layers, fine-tuning a few layers, and fine-tuning the whole model scenarios:

- **Case 1: Fine-tuning a few layers**

  If our prediction task is similar, there will be similar higher-level features or layers output. Therefore most or all of the pre-trained model layers already have relevant information about the new data set and should be kept. We will *freeze the weight of most of the starting layers* of the pre-trained model and fine-tune only the end layers.

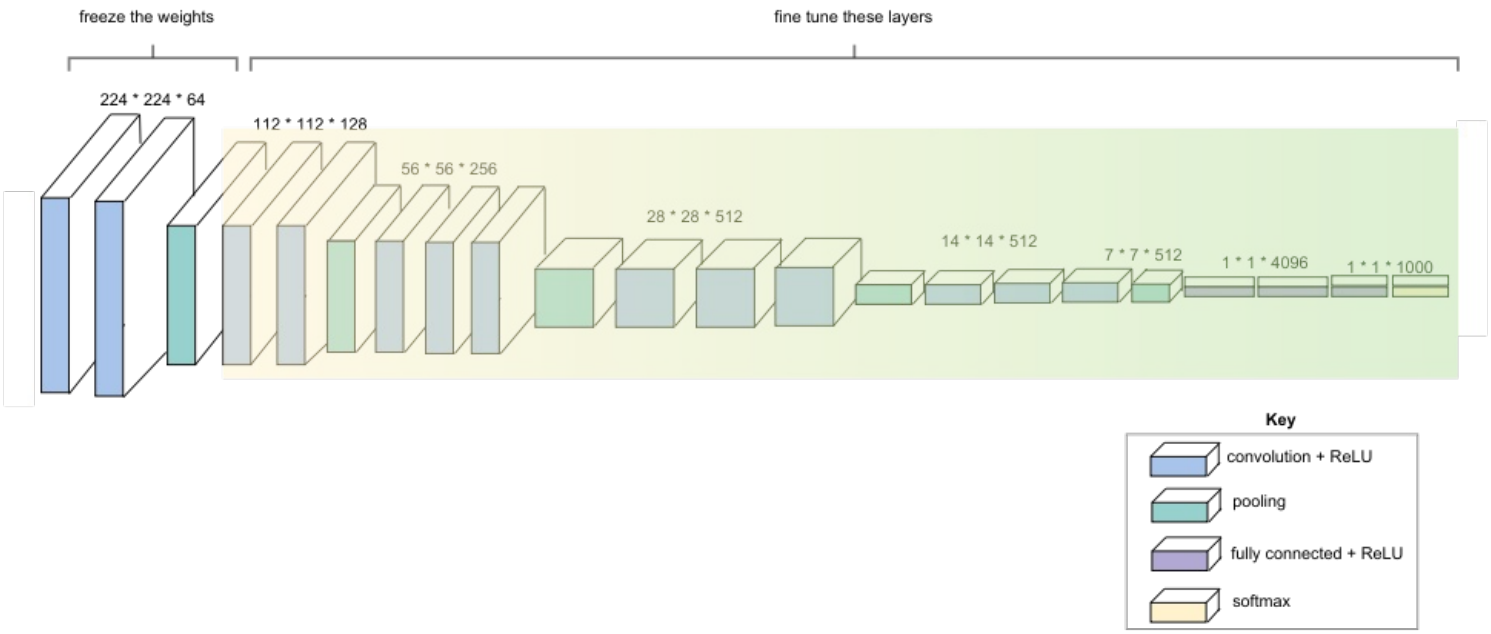
  This approach will also be most viable if our labeled training data is limited as it would be hard to re-tune all layers based on that limited data set.




Fine-tuning a few layers

- **Case 2: Fine tuning more layers**

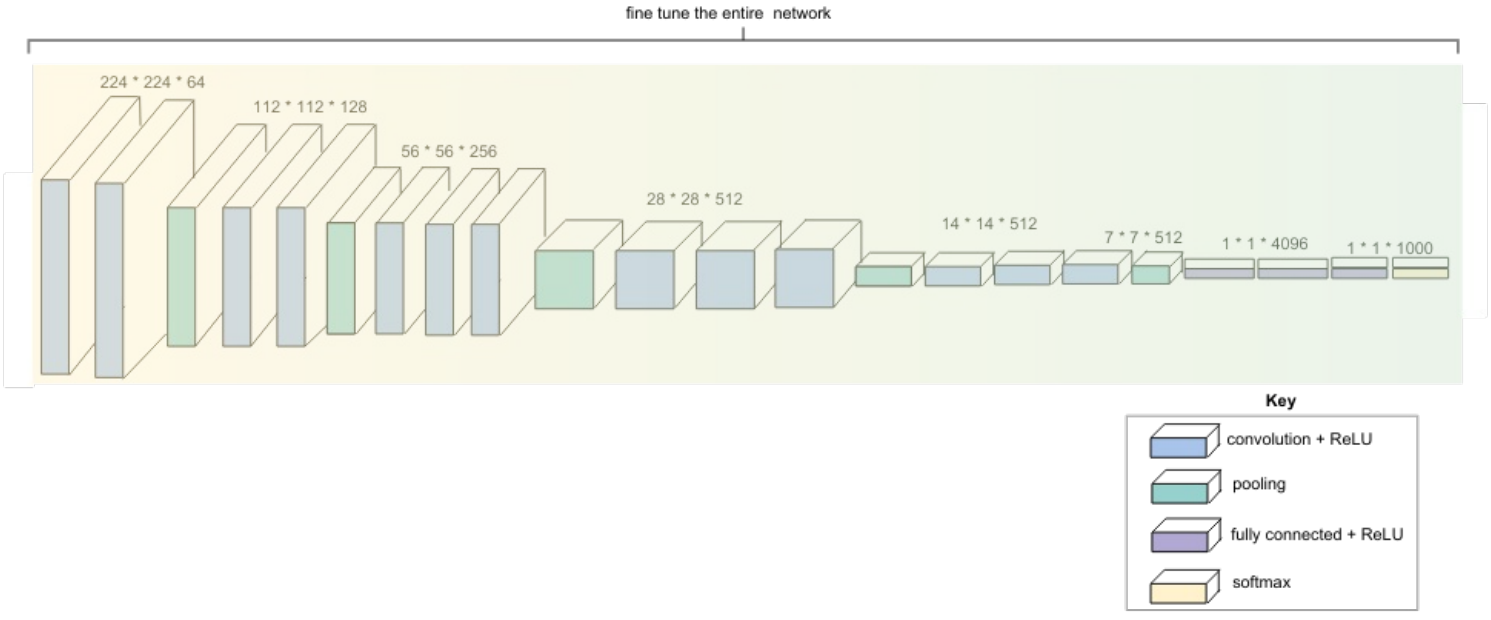
If we have significant amount of labelled examples and our learning tasks have commonalities but few differences as well, it would make sense to go deeper in fine tuning our pre-trained model. We will *freeze the weights of the first few layers* and fine-tune the weights of the remaining end layers to optimize the model for our new learning task on medical image dataset.




Fine-tuning most of the end layers

- **Case 3: Fine tuning the entire model**

  If the new data set is larger, then we will load the weights of the pre-trained model and *fine-tune the entire network.* This will definitely increase our training time as well but should help us optimize our learning task when we have significant training data.




Fine tuning the entire model

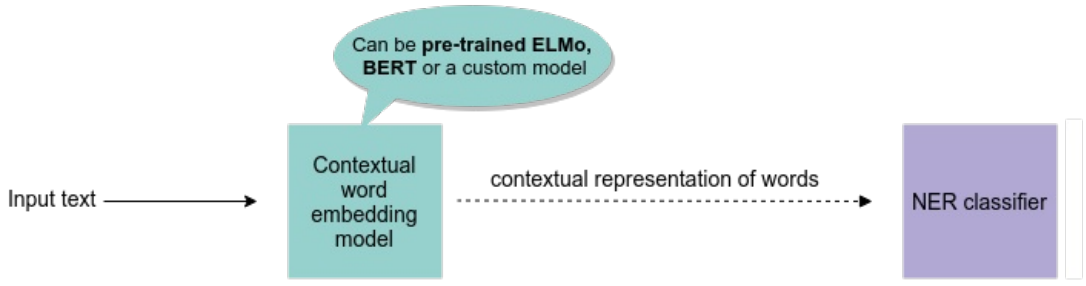
More details of transfer learning using image data are explained while modeling the solution for the image segmentation problem (https://www.educative.io/courses/grokking-the-machine-learning-interview/g77j6mXv7R9).

## Natural language processing(NLP) #

In many of NLP learning tasks such as language understanding, speech recognition, entity recognition, language generation, semantic understanding, etc. as well as other problems that are based on search, one major need is to represent our text terms in a way that they capture the semantic meaning.

For this, we need to generate the *dense representation* of textual terms. A few of the popular term representation models that use a self-supervised learning approach, trained on massive datasets are word2vec, BERT, and ELMO. The term representation based on these models capture their semantic meanings. Hence, we can transfer knowledge from this learned task in many of the NLP tasks.

Through the transfer learning approach, we can now utilize these embeddings in a NER classifier, spam detector classifier, search ranking, language understanding, etc. and can significantly improve the quality of these ML models.




BERT/ELMo can be as an embedding generator

More details can be learned about these approaches while modeling the solution for the entity linking problem (https://www.educative.io/courses/grokking-the-machine-learning-interview/YVmpGv0D01O#elmo).


← Back

Embeddings

Next →

Model Debugging and Testing

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/transfer-learning__practical-ml-techniquesconcepts__grokking-the-machine-learning-interview)