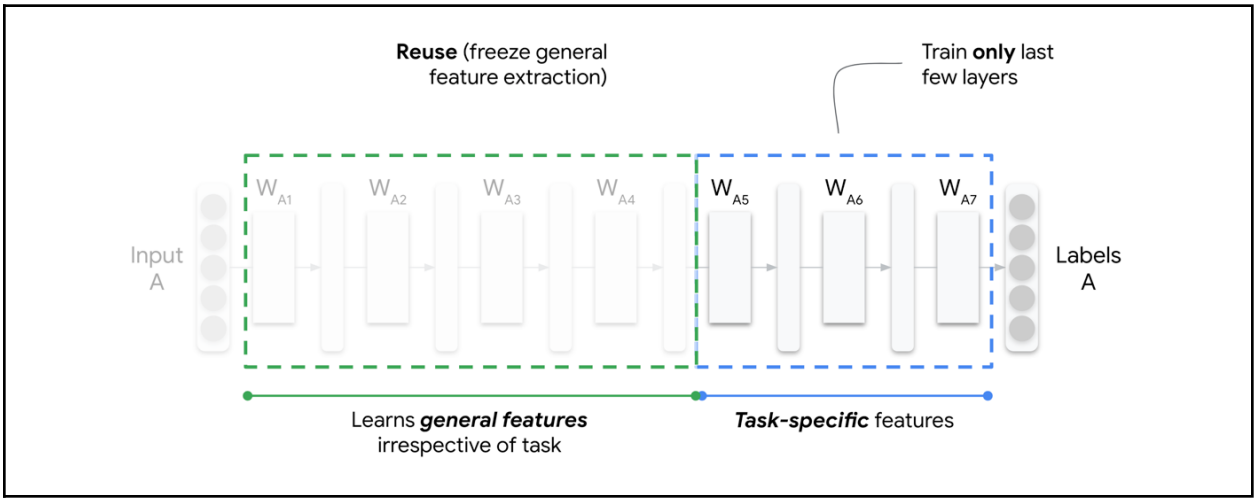
Pros and Cons of Transfer Learning

Motivation

Sometimes, we are unable to train machine learning models from scratch. This might be due to the lack of computational resources, time available, or a limited amount of available training data. Ideally, we would like to minimize unnecessary computation to help save us time, money, and also to minimize carbon emissions as a result of our computation, but also to port existing models used for general tasks to our more specific application.

Imagine a scenario where you are developing an image-captioning model to produce a description of a patient's chest x-ray. You might only have access to a small amount of data to develop this, say, 1000 x-rays and reports, and thus training an image captioning model from scratch would produce poor results. However, we can take an existing, more general, image-captioning model and use this as the starting point for our model. Sometimes, this is called "pre-initializing" or "pre-training" our network, since we import the weights from a previously trained network. Following this, we can train the latter layers of our network (while keeping the early layers frozen, since these encode more primitive image properties) on our smaller dataset of chest x-ray data. The results from this implementation should far exceed those of a model trained from scratch on the chest x-ray data. Clearly, this only works when the two tasks are similar enough that the model can be ported from the more general application to the more specific application.



To this end, repositories of models have been developed (e.g., Hugging Face) to store models that are freely available for use. Some examples of popular models are BERT, VGG16, Inception, and GPT-2. These models were developed by organizations with vast amounts of computational resources, and are thus prohibitively expensive for any individual to produce. The fact that we can use and adapt these models to our own applications is a triumph of open-source machine learning. However, despite all of the available resources, it is often difficult to tell whether transfer learning is suitable (or feasible) for a particular application.

Transfer learning is an extremely useful technique in scenarios where we have (1) limited computational resources, (2) a relatively small dataset, and (3) a specialized application of a more general task, but there are limitations to its usage as well.

Advantages

Resource Saving and Efficiency. Transfer learning allows us to take existing models that require vast amounts of computational resources, and reuse these for similar applications. This helps to save resources, reducing the associated cost, time, and power consumption of training models. As a result, efficiency is improved because resources can be diverted towards more productive activities.

Synergistic Model Performance. In situations where the initial and target model have a high degree of similarity, model performance can exceed the performance of a model trained from scratch on the transfer learning dataset. However, it should be noted that the improvement may only be marginal in some cases (or negative in the case of negative transfer) if the initial and target models are not strongly related.

Small Datasets. Transfer learning allows large neural network models to be trained with relatively little data since we are more so tuning the model as opposed to training it directly. However, generally, it is true that better performance will be achieved if a larger number of data points is used since standard error scales inversely with the number of data points.

Limitations

Negative Transfer. A major limitation of transfer learning is the problem of negative transfer. If the tasks of our initial and target models are sufficiently similar, transfer learning should provide results superior to a model that is initialized with random weights instead of the weights from a network already trained on a similar task. However, in some cases, transfer learning can lead to worse results than a model trained using random weights, which is often referred to as negative transfer. This tends to occur if the tasks are sufficiently dissimilar that the structural information encoded in the initial network is detrimental to the target model's predictions. Currently, there is no clear way to determine what the conditions are to ensure that negative transfer does not occur.

Data Suitability. While one of the benefits of transfer learning is that we can build a large machine learning model with only a small amount of training data, this is also one of the disadvantages. If our dataset is too small, and our model is too large, we run the risk of overfitting our model to the small amount of data, meaning we will achieve a poor performance at inference time. The number of data samples needed to minimize this possibility is context-dependent, but a helpful rule of thumb is that several hundred samples should be used in a transfer learning example at a minimum.

Constrained Architecture. Perhaps the most important limitation of transfer learning is that of a constrained architecture. Since we are basically importing model weights from a previously trained network, we are constrained to use the same architecture as the previous model. If we start to add or remove layers, we will likely distort the network performance to its detriment.

Transfer learning is a powerful resource to have at your disposal, but only if it makes sense for a given application and the architectural constraints are not prohibitive. Otherwise, you may have to resort to an alternative technique, such as training your model from scratch!