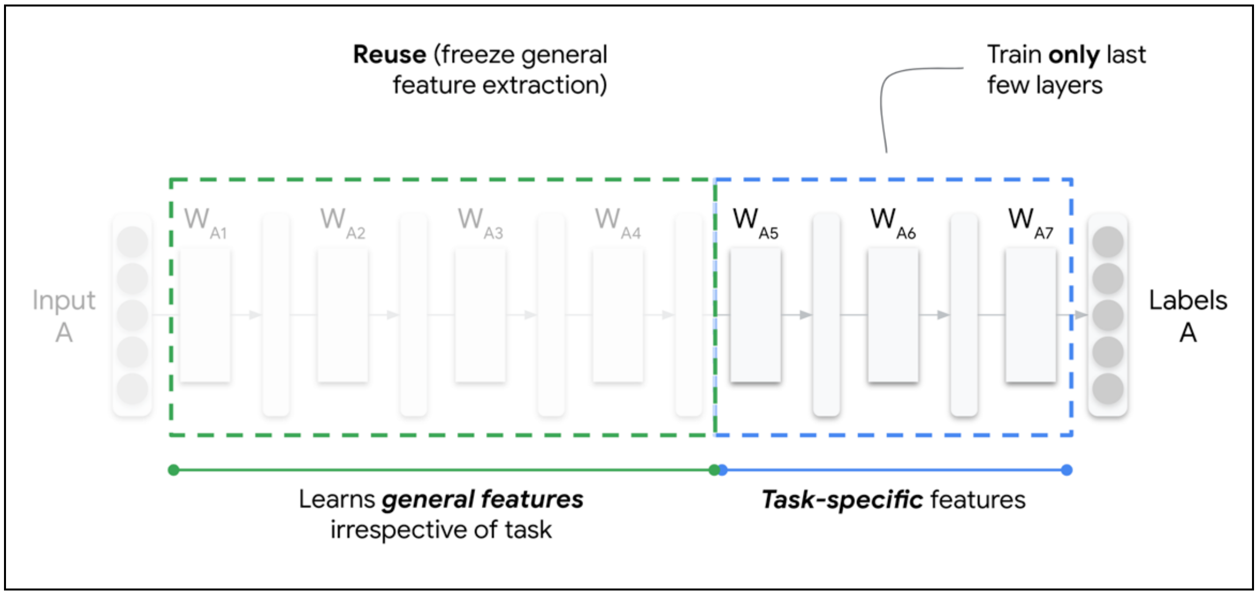
The Pitfalls of Transfer Learning


Reuse (freeze general feature extraction)
Train **only** last few layers
Input A
W_{A1}
W_{A2}
W_{A3}
W_{A4}
W_{A5}
W_{A6}
W_{A7}
Labels A
Learns ***general features*** irrespective of task
Task-specific features

Hopefully, you have now been convinced of the power of transfer learning. Transfer learning allows us to use a model generated for one task to be fine-tuned for a different but related task. In doing this, we can repurpose models for use in new tasks, thereby saving energy that would be required in computation. Furthermore, models with superior performance can be developed with a relatively small amount of data, which would normally cause overfitting in a network trained from scratch. Consequently, transfer learning provides a trifecta of benefits: reduced energy usage, faster convergence, and higher asymptotic accuracy. However, transfer learning can be difficult to implement and is not always guaranteed to be successful.

Task Similarity

Transfer learning is only viable when the task of interest is similar in scope to the original task. Image recognition is the archetypal example of this, wherein the convolutional filters close to the input can be ported to many imaging tasks due to their generality (e.g., spotting lines, edges, shapes). The further into the convolutional neural network we get, the more specific the filters become and the less transferrable they become.

Tasks that are too dissimilar may not receive any benefit from transfer learning. In fact, the use of transfer learning may even be detrimental to the final model performance. For example, performing transfer learning on a neural network used for language translation may perform well if done between two similar languages, such as two Romanic languages (e.g., French and Spanish), but may perform poorly if done between two dissimilar languages (e.g., Mandarin and French). This situation is commonly referred to as negative transfer.

Fragile Co-Adaptation

Transfer learning involves the use of neuron weights from one trained network to pre-initialize neuron weights in a second network for a related task. Often, the weights of several of the first layers are frozen so that they cannot be changed during learning. The rationale for this is to prevent noise in the new dataset from detrimentally altering the earlier layers. However, freezing weights in itself can be detrimental to model performance as it can lead to fragile co-adaptation of neurons between neighboring frozen and unfrozen layers. This occurs when neurons in the former layers are fixed and the latter layers alone are unable to adapt effectively to the new data. In practice, this is often solved by not freezing any layers and instead using a significantly smaller learning rate used when training the original network. This procedure helps to ensure former layers are not altered significantly while reducing the possibility of fragile co-adaptation.

Fixed Architecture

A key disadvantage of transfer learning is the restriction on neural architecture. Transfer learning is often performed from well-known models developed by commercial entities such as Inceptionv3, ResNet, and NASNet. These models are trained with specific neural architectures which are reflected in the model weights. Model layers cannot be modified, removed, or added without these changes propagating across the network. As such, if the network is altered, we cannot have confidence that the remaining parameters are still able to accurately model the data. This presents important limitations for model tuning and, more importantly for tiny machine learning applications, also for model size. These well-known architectures are large and often cannot be compressed sufficiently to fit within the hardware constraints of embedded systems. Thus, the use of transfer learning for tiny machine learning applications is fundamentally limited by existing model architectures unless the resources are available to train bespoke models for transfer learning using our own pre-defined architecture.

In summary, here is a general rule of thumb of when transfer learning works well when applying transfer learning to your situation where you have a *new* dataset that you wish to fine-tune your network upon.

	New dataset is similar to original dataset	*New dataset is (less) similar to original dataset*
New dataset is small	Best case scenario for transfer learning	Not the best scenario for transfer learning
New dataset is large	Transfer learning will work	Training from scratch might yield better accuracy

Despite some pitfalls, it is clear that transfer learning is a hugely powerful concept and, although difficult to implement in practice, presents many exciting opportunities for tiny machine learning applications.