Transfer Learning

What is Transfer Learning

Transfer learning allows us to take the patterns (also called weights) another model has learned from another problem and use them for our own problem.

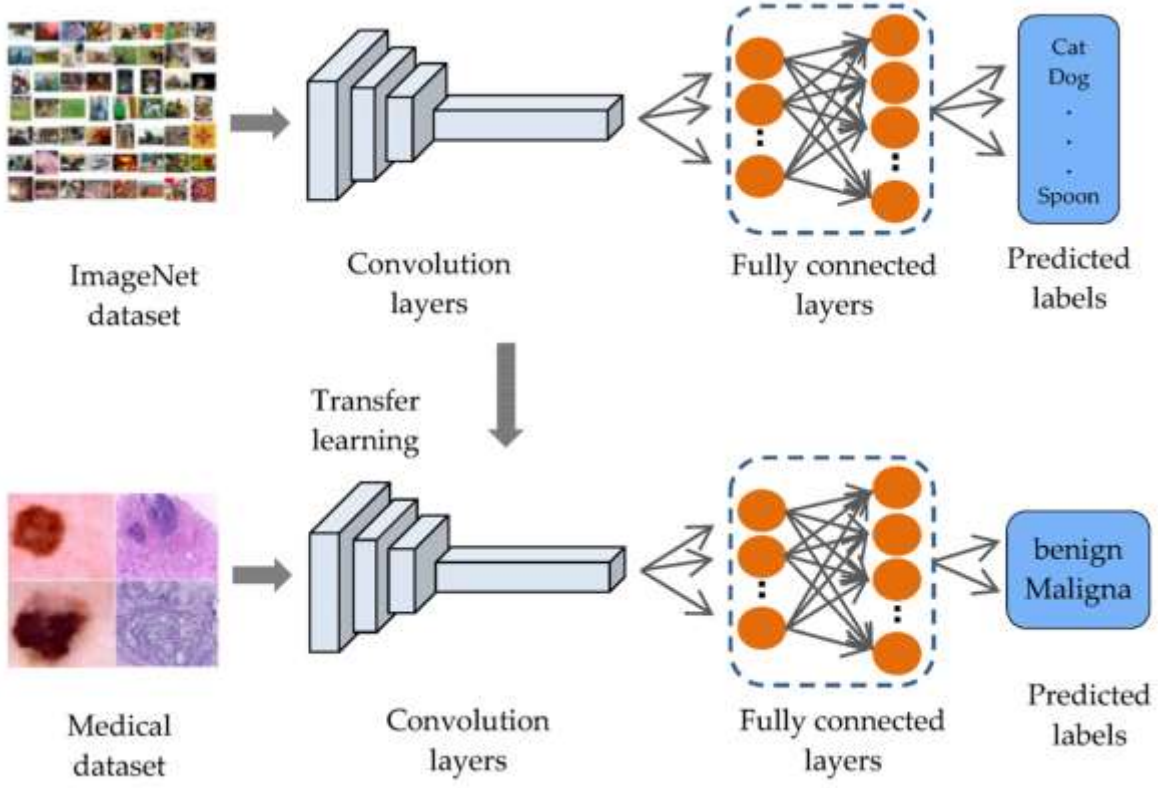
- For example, we can take the patterns a computer vision model has learned from datasets such as ImageNet (millions of images of different objects) and use them to power our custom task-specific computer vision model.
- Or we could take the patterns from a language model (a model that's been trained on a large amount of text to learn a representation of language) and use them as the basis of a refined language model specialized to our domain of interest.

Why Use Transfer Learning

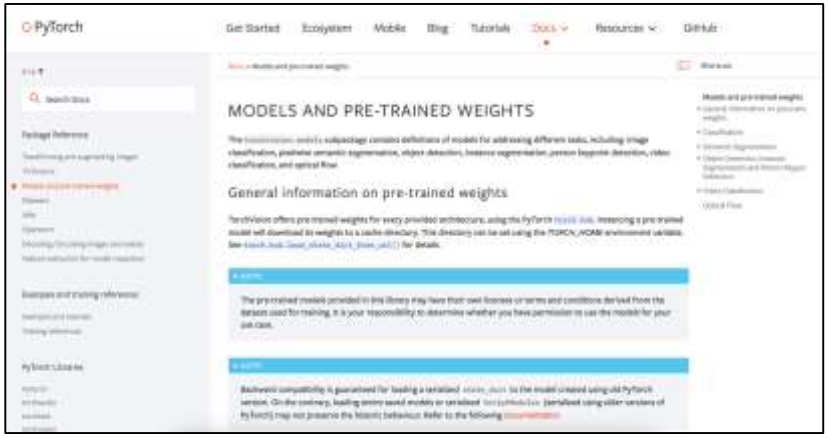
There are two main benefits to using transfer learning:

1. Can leverage an existing neural network **proven to work** on problems similar to our own.
2. Can use the representations / patterns **already learned** on similar data to our own. This often results in achieving **great results with less custom data**.

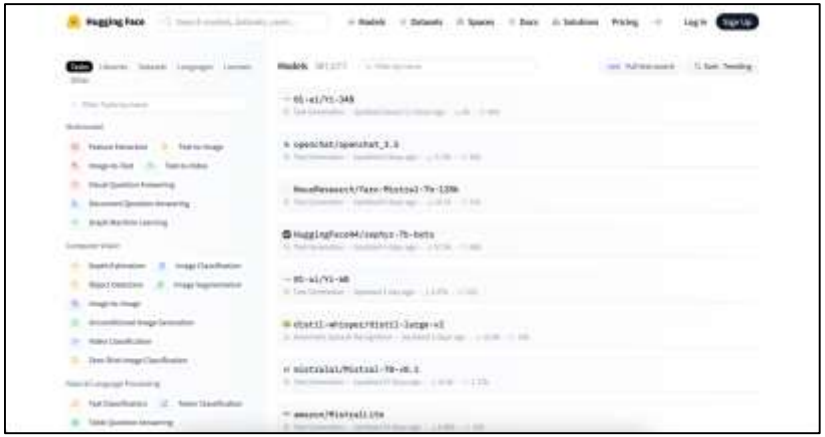
General Approach to Transfer Learning



Examples of Pre-trained Model Repositories



torchvision



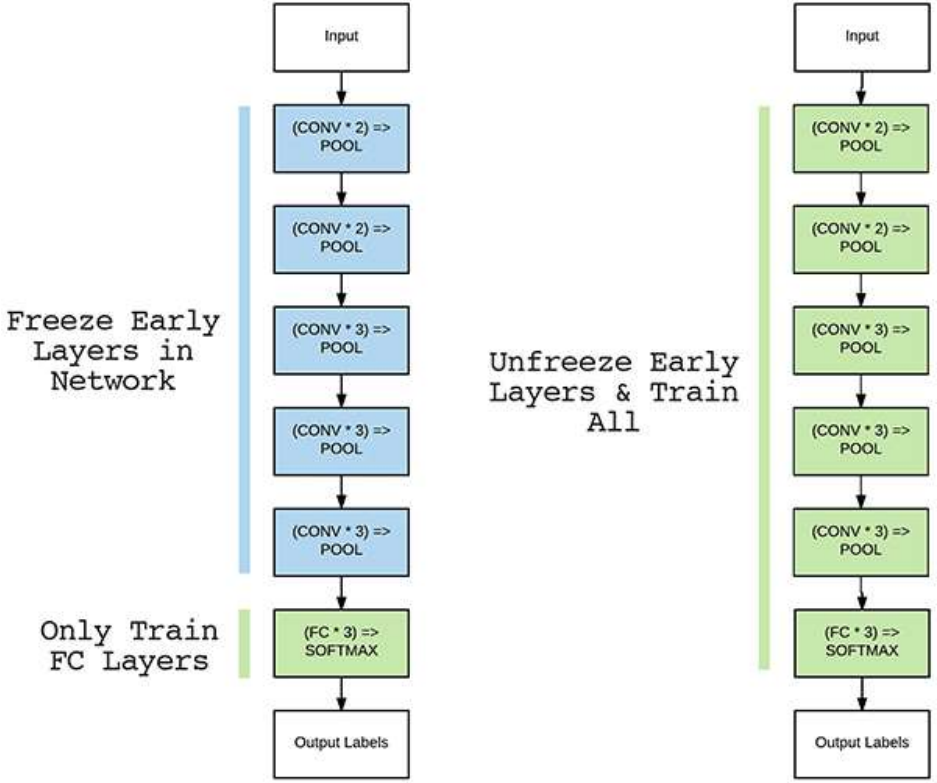
hugging face

Fine Tuning vs Feature Extraction

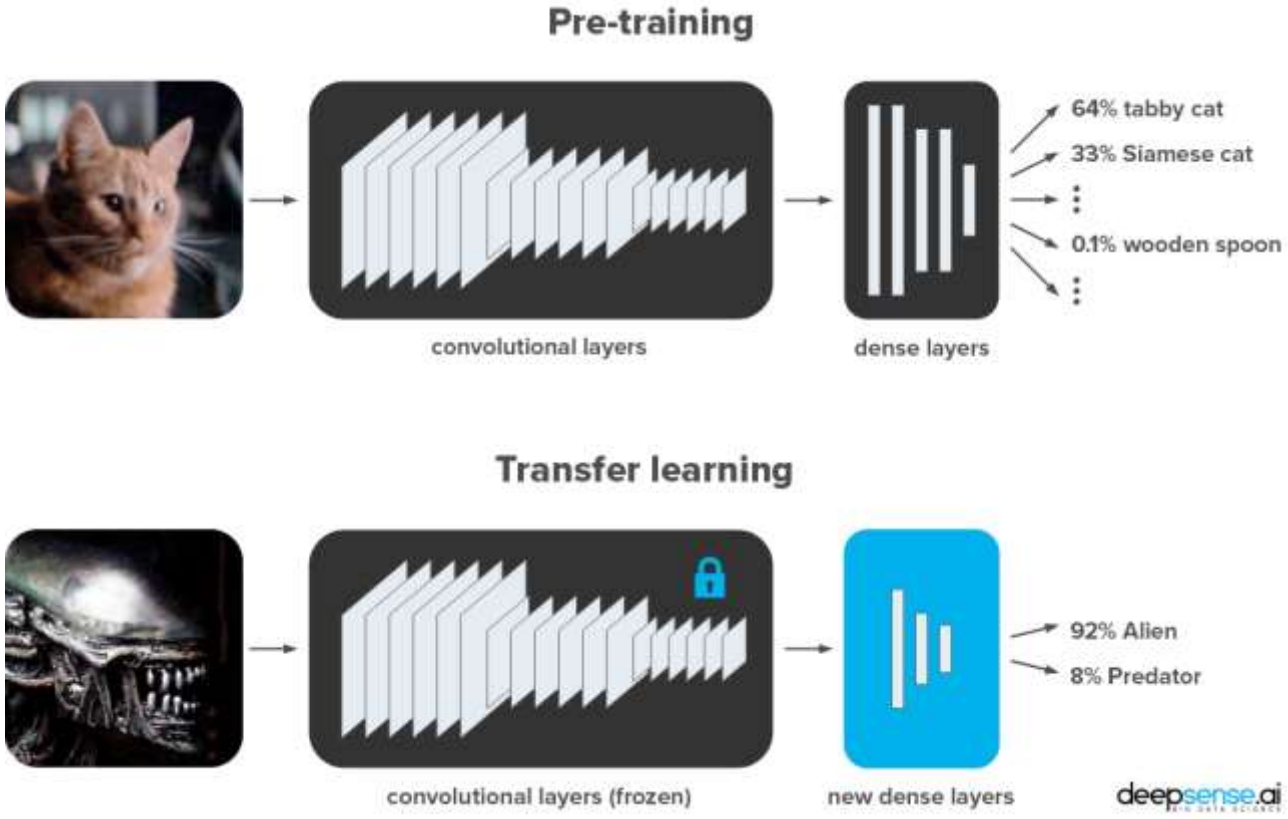
In computer vision, there are two core transfer learning scenarios:

- **Finetuning the ConvNet:** Instead of random initialization, we initialize the network with a pretrained network, like the one that is trained on imagenet 1000 dataset. Rest of the training looks as usual.
- **ConvNet as fixed feature extractor:** Here, we will freeze the weights for all of the network except that of the final fully connected layer. This last fully connected layer is replaced with a new one with random weights and only this layer is trained.

Fixed Feature Extractor vs Fine Tune Full Model



Feature Extraction



Transfer Learning in PyTorch