the idea that the choice of initial parameters for a deep neural network can have a significant regularizing effect on the model (and, to a lesser extent, that it can improve optimization). Second, it makes use of the more general idea that learning about the input distribution can help to learn about the mapping from inputs to outputs.

Both of these ideas involve many complicated interactions between several parts of the machine learning algorithm that are not entirely understood.

The first idea, that the choice of initial parameters for a deep neural network can have a strong regularizing effect on its performance, is the least well understood. At the time that pretraining became popular, it was understood as initializing the model in a location that would cause it to approach one local minimum rather than another. Today, local minima are no longer considered to be a serious problem for neural network optimization. We now know that our standard neural network training procedures usually do not arrive at a critical point of any kind. It remains possible that pretraining initializes the model in a location that would otherwise be inaccessible—for example, a region that is surrounded by areas where the cost function varies so much from one example to another that minibatches give only a very noisy estimate of the gradient, or a region surrounded by areas where the Hessian matrix is so poorly conditioned that gradient descent methods must use very small steps. However, our ability to characterize exactly what aspects of the pretrained parameters are retained during the supervised training stage is limited. This is one reason that modern approaches typically use simultaneous unsupervised learning and supervised learning rather than two sequential stages. One may also avoid struggling with these complicated ideas about how optimization in the supervised learning stage preserves information from the unsupervised learning stage by simply freezing the parameters for the feature extractors and using supervised learning only to add a classifier on top of the learned features.

The other idea, that a learning algorithm can use information learned in the unsupervised phase to perform better in the supervised learning stage, is better understood. The basic idea is that some features that are useful for the unsupervised task may also be useful for the supervised learning task. For example, if we train a generative model of images of cars and motorcycles, it will need to know about wheels, and about how many wheels should be in an image. If we are fortunate, the representation of the wheels will take on a form that is easy for the supervised learner to access. This is not yet understood at a mathematical, theoretical level, so it is not always possible to predict which tasks will benefit from unsupervised learning in this way. Many aspects of this approach are highly dependent on the specific models used. For example, if we wish to add a linear classifier on