but they are somewhat useful in the sense that examples that are actually less likely to be correctly labeled receive smaller probabilities under the model. By viewing the training set examples that are the hardest to model correctly, one can often discover problems with the way the data has been preprocessed or labeled. For example, the Street View transcription system originally had a problem where the address number detection system would crop the image too tightly and omit some of the digits. The transcription network then assigned very low probability to the correct answer on these images. Sorting the images to identify the most confident mistakes showed that there was a systematic problem with the cropping. Modifying the detection system to crop much wider images resulted in much better performance of the overall system, even though the transcription network needed to be able to process greater variation in the position and scale of the address numbers.

Reasoning about software using train and test error: It is often difficult to determine whether the underlying software is correctly implemented. Some clues can be obtained from the train and test error. If training error is low but test error is high, then it is likely that that the training procedure works correctly, and the model is overfitting for fundamental algorithmic reasons. An alternative possibility is that the test error is measured incorrectly due to a problem with saving the model after training then reloading it for test set evaluation, or if the test data was prepared differently from the training data. If both train and test error are high, then it is difficult to determine whether there is a software defect or whether the model is underfitting due to fundamental algorithmic reasons. This scenario requires further tests, described next.

Fit a tiny dataset: If you have high error on the training set, determine whether it is due to genuine underfitting or due to a software defect. Usually even small models can be guaranteed to be able fit a sufficiently small dataset. For example, a classification dataset with only one example can be fit just by setting the biases of the output layer correctly. Usually if you cannot train a classifier to correctly label a single example, an autoencoder to successfully reproduce a single example with high fidelity, or a generative model to consistently emit samples resembling a single example, there is a software defect preventing successful optimization on the training set. This test can be extended to a small dataset with few examples.

Compare back-propagated derivatives to numerical derivatives: If you are using a software framework that requires you to implement your own gradient computations, or if you are adding a new operation to a differentiation library and must define its `bprop` method, then a common source of error is implementing this gradient expression incorrectly. One way to verify that these derivatives are correct