

the image is removed. Destroying extracted features rather than original values allows the destruction process to make use of all of the knowledge about the input distribution that the model has acquired so far.

Another important aspect of dropout is that the noise is multiplicative. If the noise were additive with fixed scale, then a rectified linear hidden unit h_i with added noise ϵ could simply learn to have h_i become very large in order to make the added noise ϵ insignificant by comparison. Multiplicative noise does not allow such a pathological solution to the noise robustness problem.

Another deep learning algorithm, batch normalization, reparametrizes the model in a way that introduces both additive and multiplicative noise on the hidden units at training time. The primary purpose of batch normalization is to improve optimization, but the noise can have a regularizing effect, and sometimes makes dropout unnecessary. Batch normalization is described further in section 8.7.1.

7.13 Adversarial Training

In many cases, neural networks have begun to reach human performance when evaluated on an i.i.d. test set. It is natural therefore to wonder whether these models have obtained a true human-level understanding of these tasks. In order to probe the level of understanding a network has of the underlying task, we can search for examples that the model misclassifies. Szegedy *et al.* (2014b) found that even neural networks that perform at human level accuracy have a nearly 100% error rate on examples that are intentionally constructed by using an optimization procedure to search for an input \mathbf{x}' near a data point \mathbf{x} such that the model output is very different at \mathbf{x}' . In many cases, \mathbf{x}' can be so similar to \mathbf{x} that a human observer cannot tell the difference between the original example and the **adversarial example**, but the network can make highly different predictions. See figure 7.8 for an example.

Adversarial examples have many implications, for example, in computer security, that are beyond the scope of this chapter. However, they are interesting in the context of regularization because one can reduce the error rate on the original i.i.d. test set via **adversarial training**—training on adversarially perturbed examples from the training set (Szegedy *et al.*, 2014b; Goodfellow *et al.*, 2014b).

Goodfellow *et al.* (2014b) showed that one of the primary causes of these adversarial examples is excessive linearity. Neural networks are built out of primarily linear building blocks. In some experiments the overall function they implement proves to be highly linear as a result. These linear functions are easy