the API documentation for more details). This makes it easy to implement data augmentation for image datasets.

Another powerful technique to train very deep neural networks is to add *skip connections* (a skip connection is when you add the input of a layer to the output of a higher layer). We will explore this idea in Chapter 13 when we talk about deep residual networks.

Practical Guidelines

In this chapter, we have covered a wide range of techniques and you may be wondering which ones you should use. The configuration in Table 11-2 will work fine in most cases.

Table 11-2. Default DNN configuration

Initialization	He initialization
Activation function	ELU
Normalization	Batch Normalization
Regularization	Dropout
Optimizer	Adam
Learning rate schedule	None

Of course, you should try to reuse parts of a pretrained neural network if you can find one that solves a similar problem.

This default configuration may need to be tweaked:

- If you can't find a good learning rate (convergence was too slow, so you increased the training rate, and now convergence is fast but the network's accuracy is suboptimal), then you can try adding a learning schedule such as exponential decay.
- If your training set is a bit too small, you can implement data augmentation.
- If you need a sparse model, you can add some ℓ_1 regularization to the mix (and optionally zero out the tiny weights after training). If you need an even sparser model, you can try using FTRL instead of Adam optimization, along with ℓ_1 regularization.
- If you need a lightning-fast model at runtime, you may want to drop Batch Normalization, and possibly replace the ELU activation function with the leaky ReLU. Having a sparse model will also help.

With these guidelines, you are now ready to train very deep nets—well, if you are very patient, that is! If you use a single machine, you may have to wait for days or

even months for training to complete. In the next chapter we will discuss how to use distributed TensorFlow to train and run models across many servers and GPUs.

Exercises

1. Is it okay to initialize all the weights to the same value as long as that value is selected randomly using He initialization?
2. Is it okay to initialize the bias terms to 0?
3. Name three advantages of the ELU activation function over ReLU.
4. In which cases would you want to use each of the following activation functions: ELU, leaky ReLU (and its variants), ReLU, tanh, logistic, and softmax?
5. What may happen if you set the `momentum` hyperparameter too close to 1 (e.g., 0.99999) when using a `MomentumOptimizer`?
6. Name three ways you can produce a sparse model.
7. Does dropout slow down training? Does it slow down inference (i.e., making predictions on new instances)?
8. Deep Learning.
   a. Build a DNN with five hidden layers of 100 neurons each, He initialization, and the ELU activation function.
   b. Using Adam optimization and early stopping, try training it on MNIST but only on digits 0 to 4, as we will use transfer learning for digits 5 to 9 in the next exercise. You will need a softmax output layer with five neurons, and as always make sure to save checkpoints at regular intervals and save the final model so you can reuse it later.
   c. Tune the hyperparameters using cross-validation and see what precision you can achieve.
   d. Now try adding Batch Normalization and compare the learning curves: is it converging faster than before? Does it produce a better model?
   e. Is the model overfitting the training set? Try adding dropout to every layer and try again. Does it help?
9. Transfer learning.
   a. Create a new DNN that reuses all the pretrained hidden layers of the previous model, freezes them, and replaces the softmax output layer with a fresh new one.
   b. Train this new DNN on digits 5 to 9, using only 100 images per digit, and time how long it takes. Despite this small number of examples, can you achieve high precision?