Algorithm 7.2: Mini-batch stochastic gradient descent

Input: Training set of data points indexed by $n \in \{1, \ldots, N\}$
Batch size B
Error function per mini-batch $E_{n:n+B-1}(\mathbf{w})$
Learning rate parameter η
Initial weight vector $\mathbf{w}$
Output: Final weight vector $\mathbf{w}$

```
n ← 1
repeat
    w ← w − η∇E_{n:n+B−1}(w)  // weight vector update
    n ← n + B
    if n > N then
        shuffle data
        n ← 1
    end if
until convergence
return w
```

network in which layer l evaluates the following transformations

$$a_i^{(l)} = \sum_{j=1}^{M} w_{ij} z_j^{(l-1)} \tag{7.19}$$

$$z_i^{(l)} = \mathrm{ReLU}(a_i^{(l)}) \tag{7.20}$$

where M is the number of units that send connections to unit i, and the ReLU activation function is given by (6.17). Suppose we initialize the weights using a Gaussian $\mathcal{N}(0, \epsilon^2)$, and suppose that the outputs $z_j^{(l-1)}$ of the units in layer $l-1$ have variance λ^2. Then we can easily show that

Exercise 7.9

$$\mathbb{E}[a_i^{(l)}] = 0 \tag{7.21}$$

$$\mathrm{var}[z_j^{(l)}] = \frac{M}{2}\epsilon^2\lambda^2 \tag{7.22}$$

where the factor of $1/2$ arises from the ReLU activation function. Ideally we want to ensure that the variance of the pre-activations neither decays to zero nor grows significantly as we propagate from one layer to the next. If we therefore require that the units at layer l also have variance λ^2 then we arrive at the following choice for the standard deviation of the Gaussian used to initialize the weights that feed into a