Exercise 2.1 c

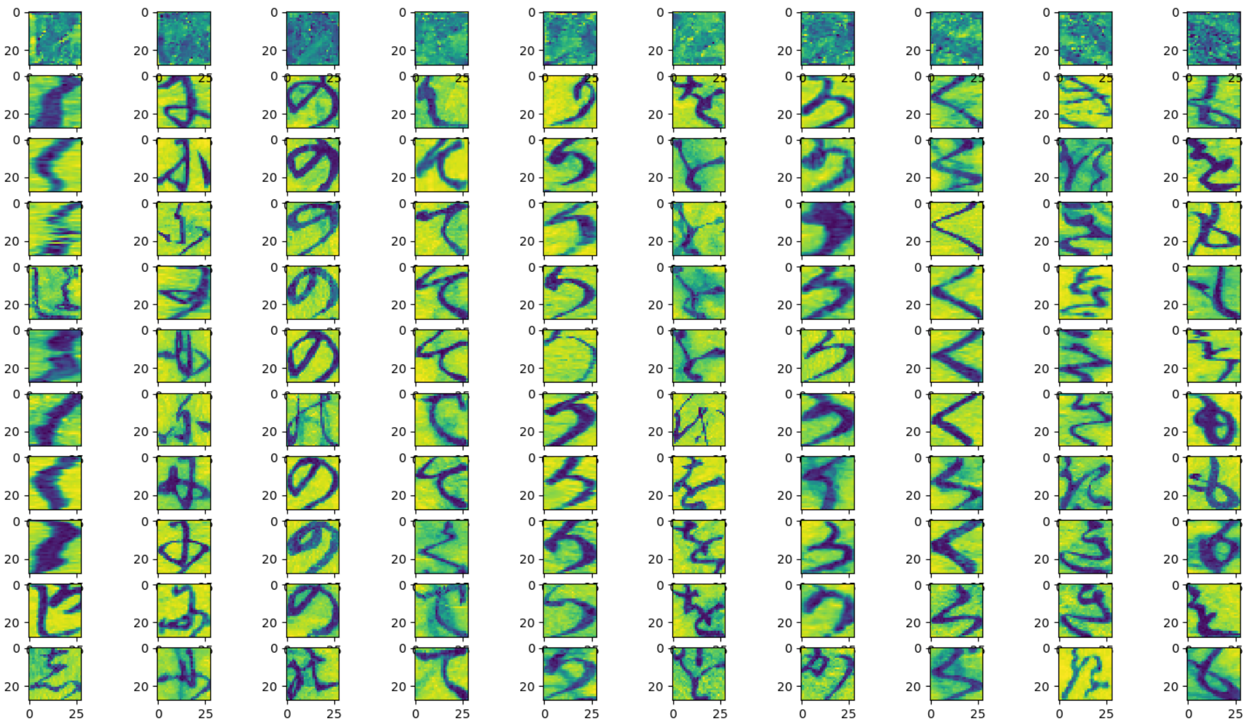

Figure 1: Visualization of the weights vectors and example data images. The darker parts in the visualization of the weights vectors look similar to the real example images.

Exercise 2.2

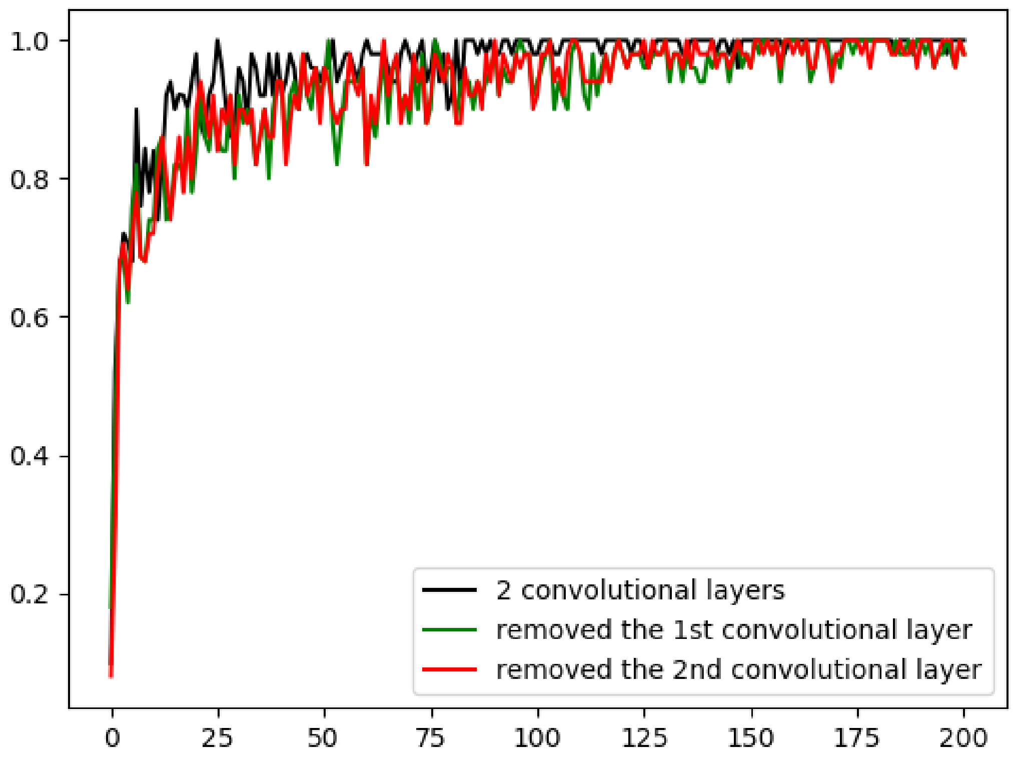

Figure 2: (d). Visualization of training accuracies while removing convolutional layers. Removing each convolutional layers increases the number of parameters. If we remove the first layer, the convolutional layer will have 64 feature maps; if we remove the second layer, it still have 64 feature maps. But no matter which layer is removed, since a max-pooling operation is also missing, the feature maps' size become twice of that with two layers and two max-pooling operations. From the figure we can see that the training accuracies with only one convolutional layer are lower than that with both layers. The test accuracy with both convolutional layer is around 97%, only with the 1st layer is around 95.1% and only with the 2nd layer is around 95.2%. Additionally, the training time only with the 2nd convolutional layer is apparently longer than only with the 1st convolutional layer.

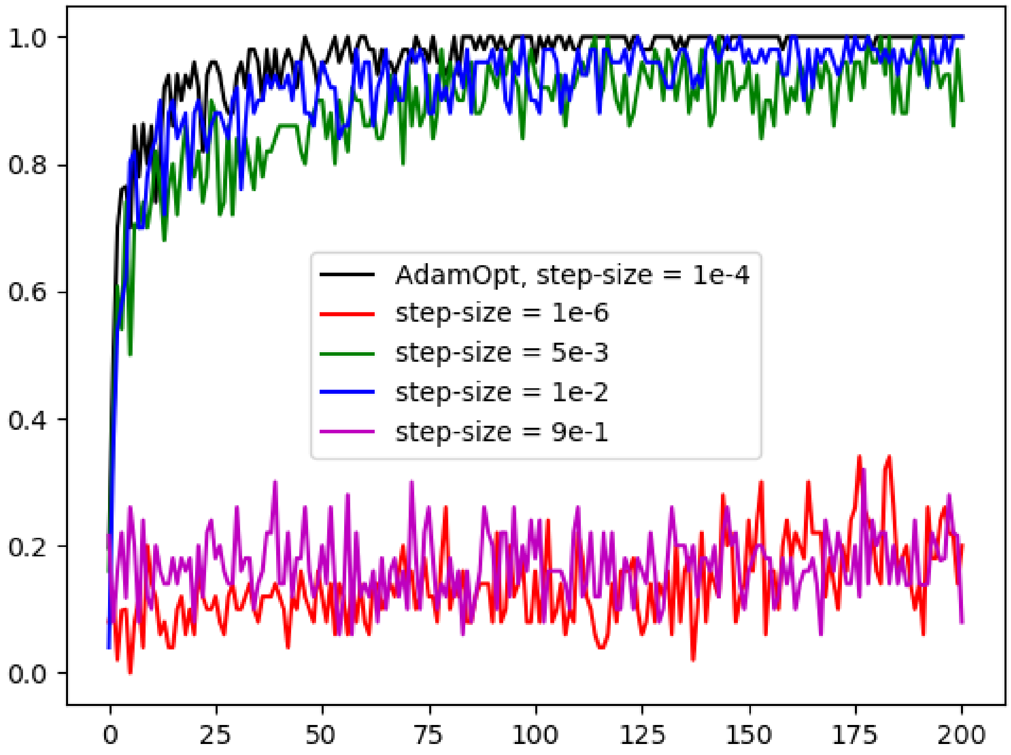

Figure 3: (e). Visualization of training accuracies with changing the step size. Too small step size will make the gradient descent progress so slowly that within the iterations it cannot reach or come out of a local minimum; Too large step size will make the gradient descent travel between two sides of a (local) minimum and never reach it.

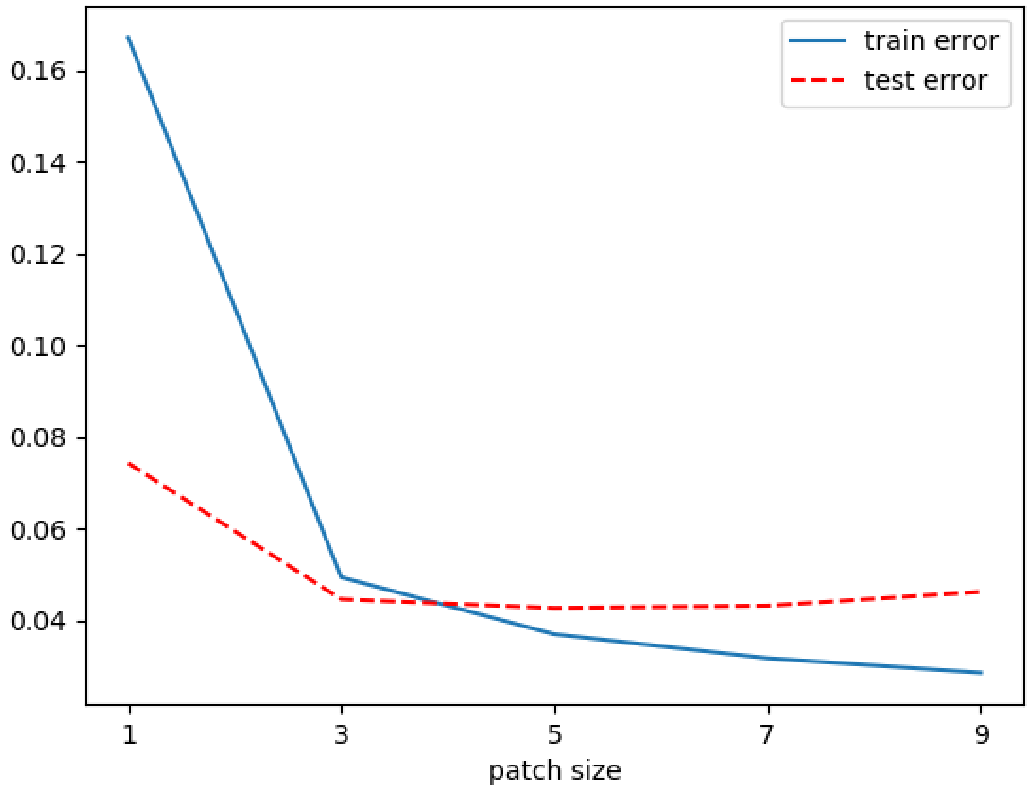

Figure 4: (f). Visualization of 5-folds cross validation training and test accuracies with changing the patch size. The patch size candidates are 1, 3, 5, 7 and 9. From the figure we could say that the patch size = 3 might be a better choice.

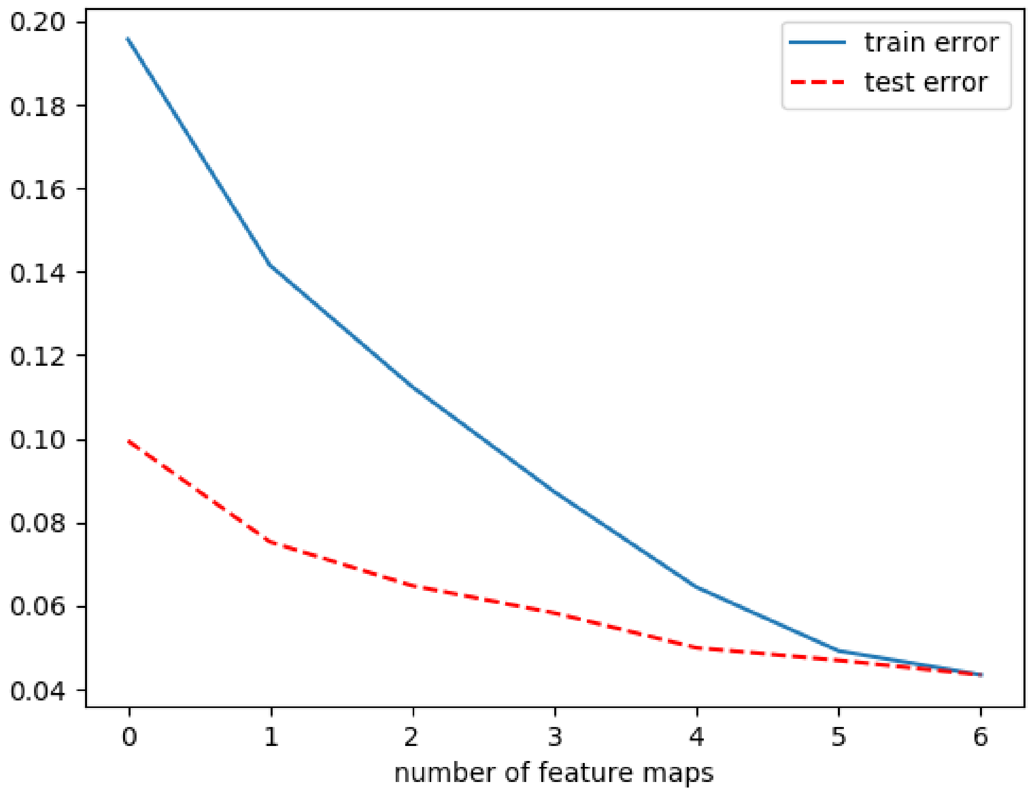

Figure 5: (f). Visualization of 5-folds cross validation training and test accuracies with changing the number of feature maps. The number of feature maps goes from 2^0 to 2^6. Assume that the 2nd layer has twice the feature maps as the 1st layer. From the figure we could see that 2^6 is exactly the turning point to over-fitting. So the kernel size of the convolutionnal layers is 3×3 and the 1st layer has 2^6 feature maps and the 2nd has 2^7.

Exercise 2.3

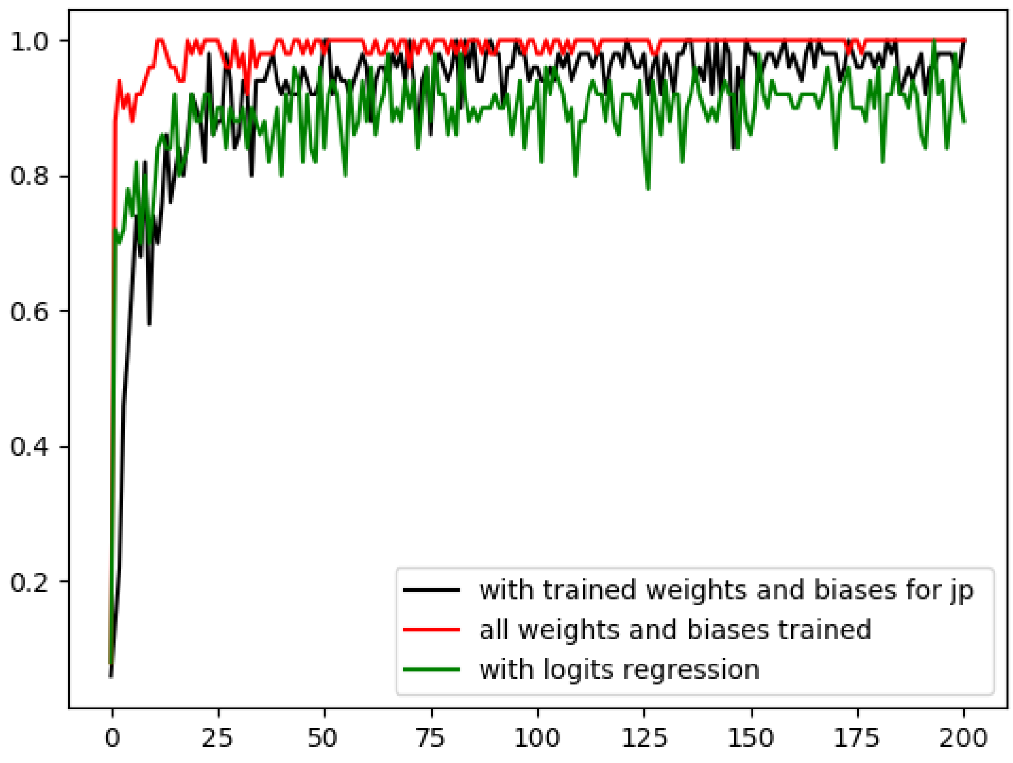

Figure 6: Visualization of training MNIST with different strategies.

Exercise 2.5

(a) $\mathbb{E}(\bar{x}) = \mathbb{E}\Big(\frac{1}{L}\sum_{l=1}^{L} x_l\Big) = \frac{1}{L}\sum_{l=1}^{L}\mathbb{E}(x_l) = \frac{1}{L}\sum_{l=1}^{L}\frac{\theta}{2} = \frac{\theta}{2}$

(b) With $\hat{x} := 2\bar{x}$ it is $\mathbb{E}(\hat{x}) = \mathbb{E}(2\bar{x}) = 2\mathbb{E}(\bar{x}) = \theta$.

(c) The standard error

$$\mathbb{E}((\hat{x}-\theta)^2) = \mathbb{E}((\hat{x}-\mathbb{E}(\hat{x}))^2) = \mathrm{Var}(\hat{x}) = \mathrm{Var}\Big(\frac{2}{n}\sum_{l=1}^{L} x_l\Big) = \frac{4}{L^2}\sum_{l=1}^{L}\mathrm{Var}(x_l) = \frac{4}{L}\mathrm{Var}(x_1)$$

clearly goes to zero for $L \to \infty$.

(d) Let $\tilde{x} := \hat{x} + \frac{1}{L}$. It is

$$\mathbb{E}(\tilde{x}) = \mathbb{E}(\hat{x} + \frac{1}{L}) = \theta + \frac{1}{L}$$

and

$$\mathbb{E}((\tilde{x}-\theta)^2) = \mathbb{E}((\hat{x} + \frac{1}{L} - \theta)^2) = \mathbb{E}((\hat{x}-\theta)^2 + \frac{1}{L^2} + \frac{2}{L}(\hat{x}-\theta)) = \frac{4}{L}\mathrm{Var}(x_1) + \frac{1}{L^2}$$

which goes to zero for $L \to \infty$.