DD2424 - Assignment 3
Bonus Point

Thi Thuy Nga Nguyen

1 Improvements

1.1 Include the omitted bias terms at each layer

I added the bias terms at each layer which are vectors of size of the number of filters for convolutional layers and a vector of size K - the number of classes (K = 18) for the output layer.

1.1.1 Gradient computation check

The numerically computed gradients, g_n, and the analytically computed gradients, g_a, were compared by using relative error

$$error = \frac{|g_a - g_n|}{max(\epsilon, |g_a| + |g_n|)},$$

where $\epsilon = 10^{-30}$. The network has 2 convolutions layers in which the first one has n_1 filters of size $(d \times k_1)$ and the second one has n_2 filters of size $(n_1 \times k_1)$. He initialization was used for the model parameters, in particular, the filters were randomly drawn from Gaussian distribution with mean 0 and standard deviation $\sqrt{\frac{2}{10 \times k_1}}$ in layer 1 and $\sqrt{\frac{2}{n_1 \times k_2}}$ in layer 2; and the weight matrices W came from a Gaussian distribution with mean 0 and standard deviation $\sqrt{\frac{2}{\text{fsize}}}$, where fsize $= n_2(n_{len} - k_1 - k2 + 2)$, n_{len} is the number of columns of the input X. Since the input is sparse I set a bound 10 instead of $d = 28$ for std of the filters in layer 1. The biases in convolutional layer were initialized from Normal distribution with standard deviation 0.0001 and 0 for output layer.
Practically, I set the model hyper-parameter as the learning rate is 0.001, the number samples of a batch is 100, momentum term is 0.9 and the network structure $k_1 = k_2 = 5$, $n_1 = n_2 = 5$ and did the gradient computation check for first 6 names in dataset and $h = 1e - 5$, using mini-batch of size 1, I got

```
The number of error (relative error > 1e-6) of W: 129
The maximum of relative error of W: 8.313597449953456e-05
The number of error (relative error > 1e-6) of b3: 0
```

```
The maximum of relative error of b3: 4.980173922193425e-09
The number of errors (relative error > 1e-6) of filter 0: 0
The maximum of relative error of filter 0: 1.921156143509407e-07
The number of errors (relative error > 1e-6) of bias 0: 0
The maximum of relative error of b 0: 2.997694343263668e-09
The number of errors (relative error > 1e-6) of filter 1: 0
The maximum of relative error of filter 1: 9.630391294521692e-07
The number of errors (relative error > 1e-6) of bias 1: 0
The maximum of relative error of b 1: 5.823431880928594e-10
```

When using mini-batch of size 2, I got

```
The number of error (relative error > 1e-6) of W: 122
The maximum of relative error of W: 0.0069560836058083315
The number of error (relative error > 1e-6) of b3: 0
The maximum of relative error of b3: 7.974742817317513e-07
The number of errors (relative error > 1e-6) of filter 0: 1
The maximum of relative error of filter 0: 1.6165800171338403e-06
The number of errors (relative error > 1e-6) of bias 0: 0
The maximum of relative error of b 0: 2.9727093812031007e-10
The number of errors (relative error > 1e-6) of filter 1: 0
The maximum of relative error of filter 1: 1.1562897341324123e-07
The number of errors (relative error > 1e-6) of bias 1: 0
The maximum of relative error of b 1: 1.1795903428493475e-08
```

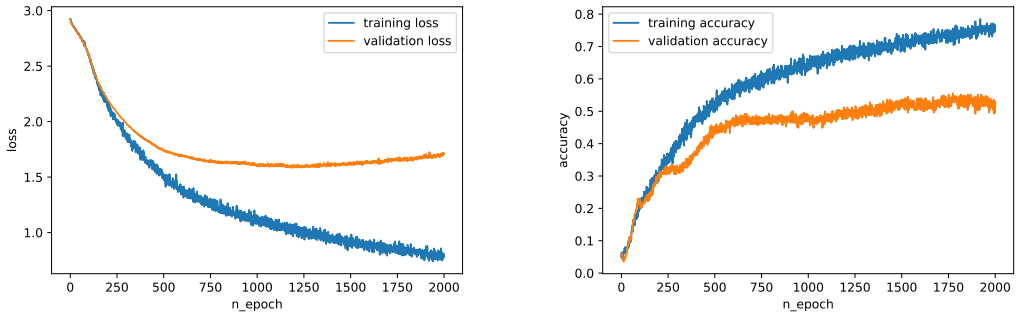
And using mini-batch of size 3, I got

```
The number of error (relative error > 1e-6) of W: 0
The maximum of relative error of W: 4.2298077869674004e-07
The number of error (relative error > 1e-6) of b3: 0
The maximum of relative error of b3: 7.790466723474956e-09
The number of errors (relative error > 1e-6) of filter 0: 0
The maximum of relative error of filter 0: 1.2791852702036003e-07
The number of errors (relative error > 1e-6) of bias 0: 0
The maximum of relative error of b 0: 2.16129599406516e-09
The number of errors (relative error > 1e-6) of filter 1: 0
The maximum of relative error of filter 1: 5.298490823686815e-08
The number of errors (relative error > 1e-6) of bias 1: 0
The maximum of relative error of b 1: 1.2830114851554289e-09
```

The relative errors being almost very small show the precision for the gradients computation.

1.1.2 Train CNN with bias terms

With an initialization of 0 for all bias terms, I trained the network of 2 convolutional layers with setting as above for 2000 epochs (corresponding to 36000 updates) with compensating for unbalance training data and got 52.78% of the final validation accuracy which is not an improvement comparing to CNN without biases which gave the same final accuracy after same training epochs. However, the network looks like overfitting (figure 1). At the end the validation accuracy goes down. The best performance is 54.37% after 1900 epochs.



(a) (b)

Figure 1: The loss (a) and the accuracy (b) after each epoch (18 updates) on the validation set with $\eta = 0.001, n\_batch = 59$, and $n\_epochs = 2000$.

1.2 Networks with L convolutional layers

I made a search for the good number of convolutional layers and the good number of filters in each convolutional layer. Note that in this part I did on CNN without bias terms. In practice, I tried with (2,3,4) convolutional layers which are the same number of filters varying in (5,10,15,20). The size of the first convolutional layer is always 5 and of the others is 3. After training 200 epochs (3600 updates) I got the table 1 of the validation accuracies and table 2 after 500 epochs.

The number of filters	2 conv layers	3 conv layers	4 conv layers
5	14.68%	13.89%	12.7%
10	24.6%	22.22%	21.83%
15	22.62%	26.59%	22.22%
20	25.79%	28.17%	27.38%

Table 1: The validation accuracies of CNNs after 200 epochs.

The number of filters	2 conv layers	3 conv layers	4 conv layers
5	32.54%	25.4%	30.56%
10	33.33%	32.94%	38.89%
15	38.89%	39.68%	40.48%
20	42.46%	43.65%	37.7%

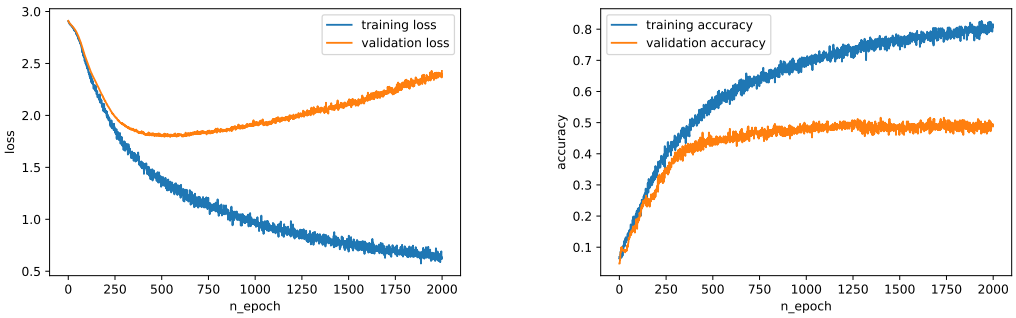
Table 2: The validation accuracies of CNNs after 500 epochs.

Note that using more filters it spends longer time in executing.

We can see that, increasing the number of layers improved the performance but not too much.

2 Four-layer CNN

I trained a CNN of 3 convolutional layers and one output layer with bias terms for 2000 epochs (36000 updates). The conv layers had (20, 20, 20) filters of size (5, 3, 3) respectively. With the same hyper-parameters described above and n_batch=59 I got 48.81% of final validation accuracy which is showed in figure 2. This performance is worse than the network with 2 convolutional layer. Moreover, the loss of validation set goes up after 500 epochs.



(a) (b)

Figure 2: The validation loss (a) and accuracy (b) at each epoch (18 updates) after training CNN with 3 convolutional layers, $\eta = 0.001, n\_batch = 59$, and $n\_epochs = 2000$.