**Assignment 3**

Yaoding Chen

**Notebooks:**

* **Q1:** <https://colab.research.google.com/drive/1hCUuTwKADArfOAC12neiauQMIeSuadV2?usp=sharing>
* **Q2a training:** <https://colab.research.google.com/drive/1MxuBQ0RPlqkP75eJJdiVybbupCtaSaVZ?usp=sharing>
* **Q2b training:** <https://colab.research.google.com/drive/1CAynf8nD-nFv5rQLXgenrK9qjfqzYsg_?usp=sharing>
* **Q2 comparison:** <https://colab.research.google.com/drive/1i2C_mNQOjoIfKodu8IGEaoWoHZ3prgXO?usp=sharing>

**Question 1. Implementation of deep learning algorithms.**

Please read the code in the Jupyter Notebook attached. [[1]](#footnote-1)

**Question 2. Training image classifier with and without batch normalization.**

Here we construct two models as follows:

|  |  |
| --- | --- |
| **Q2a. Without BN** | **Q2. With BN** |
| Model: "sequential\_2"  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =================================================================  conv2d\_16 (Conv2D) (None, 32, 32, 16) 448    conv2d\_17 (Conv2D) (None, 32, 32, 16) 2320    max\_pooling2d\_6 (MaxPooling (None, 16, 16, 16) 0  2D)    conv2d\_18 (Conv2D) (None, 16, 16, 32) 4640    conv2d\_19 (Conv2D) (None, 16, 16, 32) 9248    max\_pooling2d\_7 (MaxPooling (None, 8, 8, 32) 0  2D)    conv2d\_20 (Conv2D) (None, 8, 8, 64) 18496    conv2d\_21 (Conv2D) (None, 8, 8, 64) 36928    max\_pooling2d\_8 (MaxPooling (None, 4, 4, 64) 0  2D)    conv2d\_22 (Conv2D) (None, 4, 4, 128) 73856    conv2d\_23 (Conv2D) (None, 4, 4, 128) 147584    average\_pooling2d\_2 (Averag (None, 1, 1, 128) 0  ePooling2D)    flatten\_2 (Flatten) (None, 128) 0    dense\_2 (Dense) (None, 10) 1290    =================================================================  Total params: 294,810  Trainable params: 294,810  Non-trainable params: 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Model: "sequential\_3"  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =================================================================  conv2d\_24 (Conv2D) (None, 32, 32, 16) 448    batch\_normalization\_8 (Batc (None, 32, 32, 16) 64  hNormalization)    conv2d\_25 (Conv2D) (None, 32, 32, 16) 2320    batch\_normalization\_9 (Batc (None, 32, 32, 16) 64  hNormalization)    max\_pooling2d\_9 (MaxPooling (None, 16, 16, 16) 0  2D)    conv2d\_26 (Conv2D) (None, 16, 16, 32) 4640    batch\_normalization\_10 (Bat (None, 16, 16, 32) 128  chNormalization)    conv2d\_27 (Conv2D) (None, 16, 16, 32) 9248    batch\_normalization\_11 (Bat (None, 16, 16, 32) 128  chNormalization)    max\_pooling2d\_10 (MaxPoolin (None, 8, 8, 32) 0  g2D)    conv2d\_28 (Conv2D) (None, 8, 8, 64) 18496    batch\_normalization\_12 (Bat (None, 8, 8, 64) 256  chNormalization)    conv2d\_29 (Conv2D) (None, 8, 8, 64) 36928    batch\_normalization\_13 (Bat (None, 8, 8, 64) 256  chNormalization)    max\_pooling2d\_11 (MaxPoolin (None, 4, 4, 64) 0  g2D)    conv2d\_30 (Conv2D) (None, 4, 4, 128) 73856    batch\_normalization\_14 (Bat (None, 4, 4, 128) 512  chNormalization)    conv2d\_31 (Conv2D) (None, 4, 4, 128) 147584    batch\_normalization\_15 (Bat (None, 4, 4, 128) 512  chNormalization)    average\_pooling2d\_3 (Averag (None, 1, 1, 128) 0  ePooling2D)    flatten\_3 (Flatten) (None, 128) 0    dense\_3 (Dense) (None, 10) 1290    =================================================================  Total params: 296,730  Trainable params: 295,770  Non-trainable params: 960  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  <matplotlib.image.AxesImage at 0x7f6e4d759cd0> |

|  |  |
| --- | --- |
| Model without BN, original data    Model without BN, augmented data | Model with BN, original data    Model with BN, augmented data |

After learning rate and other hyperparameter tuning, the accuracy of the model in the training set is close to 100% and the validation accuracy stabilizes at 76.36% and 79.58% respectively. Further improvement to the models is extremely hard unless the dataset is augmented. Still, in both cases, the models with BN have higher accuracy than those without BN.

The training details are available at the two attached Jupyter Notebooks.[[2]](#footnote-2)[[3]](#footnote-3)

Below we show the feature maps of the average activation of layers, which may help to explain what contribute to our observations. The right one is using batch normalization, which shows a much faster learning rate. In CNN, since the filters to each layer are shared across feature map, normalization of the output is thus shared across the feature map. This has three effects on learning:

* First, normalization and standardization can make training faster. This is consistent with our training results, where the model with BN has much faster learning rate, as shown on the figures above.
* Second, Batch Norm reduces the internal covariate shift of the network. Thus, the amount of change in the distribution of the input of layers is reduced.
* Third, it seems that Batch Norm has a regularization effect thanks to minibatches that regularizes the loss, thus reducing the effect of noise. This explains much faster and more accurate learning we see in the following feature maps. The code to achieve this is at [[4]](#footnote-4).

**Mode without BN (left) vs model without BN (right)**

|  |  |
| --- | --- |
| Background pattern  Description automatically generated | Background pattern  Description automatically generated |

|  |  |
| --- | --- |
|  |  |
|  |  |

1. <https://colab.research.google.com/drive/1hCUuTwKADArfOAC12neiauQMIeSuadV2?usp=sharing> [↑](#footnote-ref-1)
2. <https://colab.research.google.com/drive/1MxuBQ0RPlqkP75eJJdiVybbupCtaSaVZ?usp=sharing> [↑](#footnote-ref-2)
3. <https://colab.research.google.com/drive/1CAynf8nD-nFv5rQLXgenrK9qjfqzYsg_?usp=sharing> [↑](#footnote-ref-3)
4. <https://colab.research.google.com/drive/1i2C_mNQOjoIfKodu8IGEaoWoHZ3prgXO?usp=sharing> [↑](#footnote-ref-4)