IEOR E4742 Assignment 2

Name: Mark Ma UNI: km4054

Nov 7, 2024

Problem 1 (Convolutional Neural Networks)

In the sample code `example_CNN_CIFAR.ipynb`:

(a) **Add one more convolutional layer with max pooling and assess the impact of the additional convolutional layer on accuracy.**

(b) **What is the number of parameters we are trying to learn in the original code, and how does it change with the extra layer?**

Solution to (a)

New CNN model

- **Input Layer:** Shape (32, 32, 3)
- **Convolutional Block 1:**
  - Conv2D (32 filters, 3x3 kernel, ReLU, padding='same')
  - MaxPooling2D (2x2)
- **Convolutional Block 2:**
  - Conv2D (64 filters, 3x3 kernel, ReLU, padding='same')
  - MaxPooling2D (2x2)
- **Convolutional Block 3:**
  - Conv2D (128 filters, 3x3 kernel, ReLU, padding='same')
  - MaxPooling2D (2x2)
- **Additional Convolutional Block:**
  - **New Layer:** Conv2D (256 filters, 3x3 kernel, ReLU, padding='same')
  - MaxPooling2D (2x2)
- **Flatten Layer**

- **Dense Layer:** 128 units, ReLU
- **Dense Layer:** 64 units, ReLU
- **Output Layer:** Dense, 10 units (softmax for classification)

Results

After adding the additional layer, we trained the model for 20 epochs with a batch size of 64. The loss and accuracy for both training and validation in original and new model settings are suggested follow:

- **Training:**
  - **Original Model:** The original model achieved an accuracy of 0.9459 and a loss of 0.1290.
  - **New Model:** The new model improved its training accuracy to 0.9698 and reduced the loss to 0.0790.

  Explanation: The new model demonstrated enhanced learning ability on the training set, likely due to the added layer, which increased the model's capacity to fit the training data more effectively.

- **Validation:**
  - **Original Model:** The original model's validation accuracy was 0.7231, with a loss of 1.4876.
  - **New Model:** The new model showed a slightly lower validation accuracy of 0.7336 and a higher validation loss of 1.3954.

  Explanation: Despite the better training performance, the new model's decrease in validation accuracy and increase in loss suggest potential overfitting, where the model learned the training data too well but struggled to generalize to unseen data.

The plots of both loss and accuracy in the original model and new model are shown as Figure 1 and Figure 2.

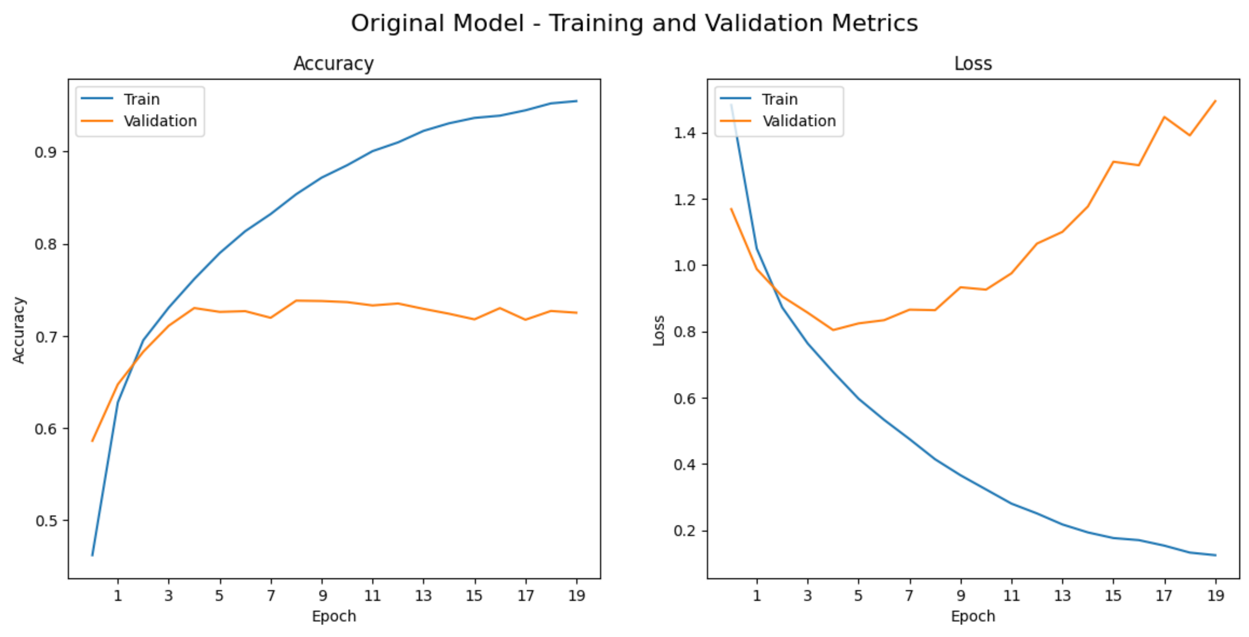


Figure 1: Training and Validation Metrics for Original Model

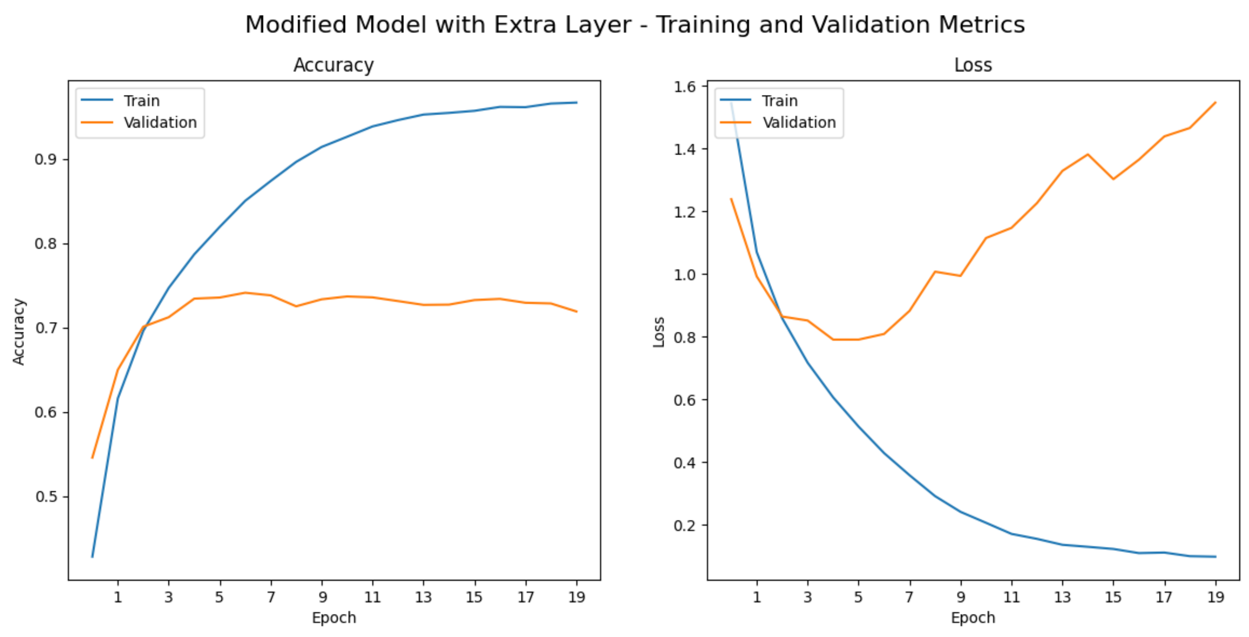


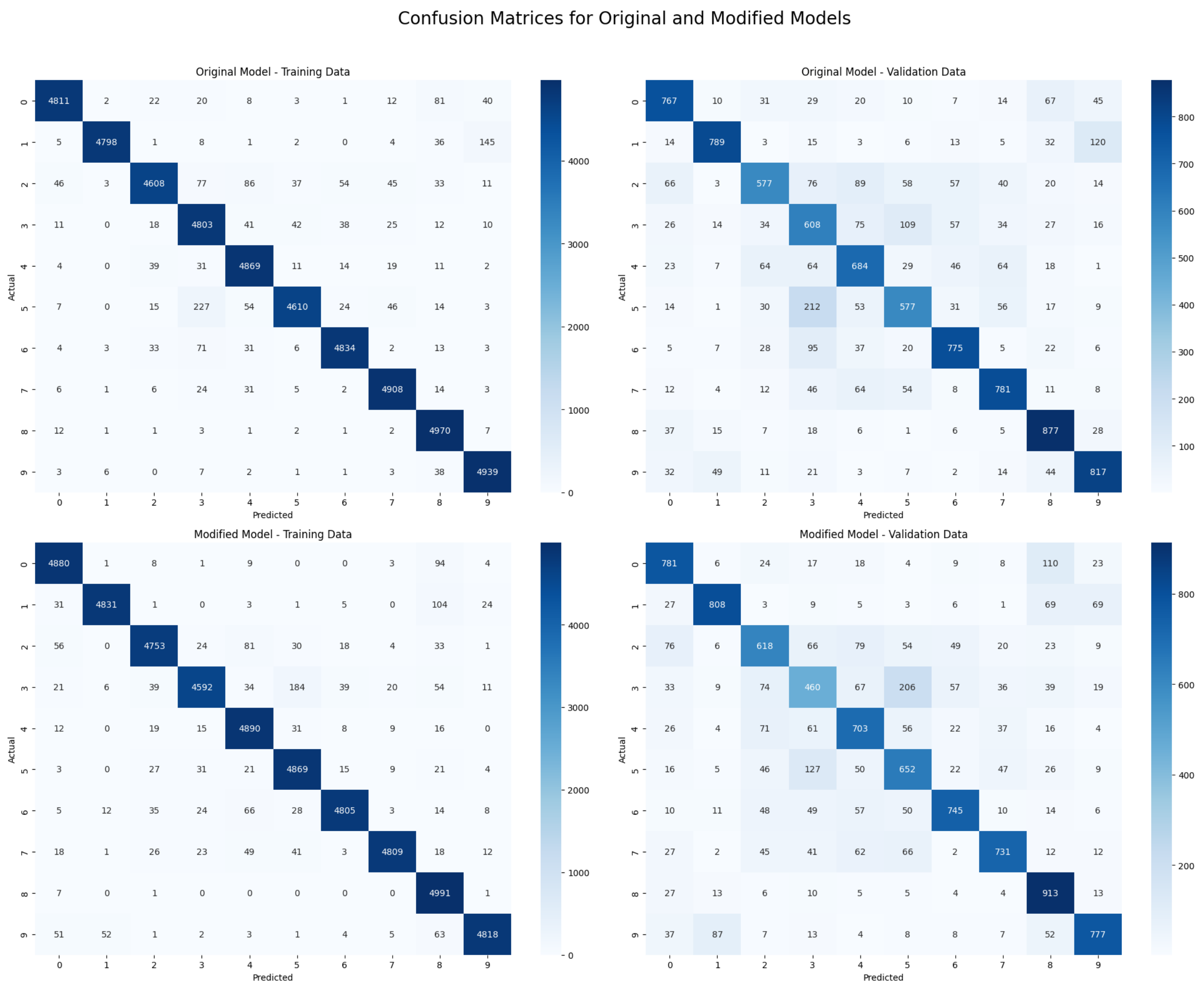
Figure 2: Training and Validation Metrics for New Model

In summary, New model performs better in the training phase, but they're quite similar in the result of validation phase.

Then, we have the confusion matrix plots, which give us more information on how the samples are classified.

- **Training:** The Modified Model outperforms the Original Model in training data classification, showing fewer misclassifications and higher accuracy. This suggests that the changes made in the Modified Model enhanced its capacity to learn from the training data more effectively.

- **Validation:** On the validation data, both the Original and Modified Models exhibit similar performance, with comparable confusion matrix patterns. This indicates that the improvements seen in training did not result in significantly better generalization for unseen data.



Original Model - Training Data

	0	1	2	3	4	5	6	7	8	9
0	4811	2	22	20	8	3	1	12	81	40
1	5	4798	1	8	1	2	0	4	36	145
2	46	3	4608	77	86	37	54	45	33	11
3	11	0	18	4803	41	42	38	25	12	10
4	4	0	39	31	4869	11	14	19	11	2
5	7	0	15	227	54	4610	24	46	14	3
6	4	3	33	71	31	6	4834	2	13	3
7	6	1	6	24	31	5	2	4908	14	3
8	12	1	1	3	1	2	1	2	4970	7
9	3	6	0	7	2	1	1	3	38	4939

Original Model - Validation Data

	0	1	2	3	4	5	6	7	8	9
0	767	10	31	29	20	10	7	14	67	45
1	14	789	3	15	3	6	13	5	32	120
2	66	3	577	76	89	58	57	40	20	14
3	26	14	34	608	75	109	57	34	27	16
4	23	7	64	64	684	29	46	64	18	1
5	14	1	30	212	53	577	31	56	17	9
6	5	7	28	95	37	20	775	5	22	6
7	12	4	12	46	64	54	8	781	11	8
8	37	15	7	18	6	1	6	5	877	28
9	32	49	11	21	3	7	2	14	44	817

Modified Model - Training Data

	0	1	2	3	4	5	6	7	8	9
0	4880	1	8	1	9	0	0	3	94	4
1	31	4831	1	0	3	1	5	0	104	24
2	56	0	4753	24	81	30	18	4	33	1
3	21	6	39	4592	34	184	39	20	54	11
4	12	0	19	15	4890	31	8	9	16	0
5	3	0	27	31	21	4869	15	9	21	4
6	5	12	35	24	66	28	4805	3	14	8
7	18	1	26	23	49	41	3	4809	18	12
8	7	0	1	0	0	0	0	0	4991	1
9	51	52	1	2	3	1	4	5	63	4818

Modified Model - Validation Data

	0	1	2	3	4	5	6	7	8	9
0	781	6	24	17	18	4	9	8	110	23
1	27	808	3	9	5	3	6	1	69	69
2	76	6	618	66	79	54	49	20	23	9
3	33	9	74	460	67	206	57	36	39	19
4	26	4	71	61	703	56	22	37	16	4
5	16	5	46	127	50	652	22	47	26	9
6	10	11	48	49	57	50	745	10	14	6
7	27	2	45	41	62	66	2	731	12	12
8	27	13	6	10	5	5	4	4	913	13
9	37	87	7	13	4	8	8	7	52	777

Figure 3: Confusion Matrix

Overall, while the Modified Model fits the training data better, both models show comparable effectiveness when tested on unseen data.

Solution to (b)

Detailed Parameter Calculation for Each Layer

Original Model

- **Conv2D (32 filters, input shape (32, 32, 3))**:

$$(3 \times 3 \times 3 + 1) \times 32 = (27 + 1) \times 32 = 896 \text{ parameters}$$

- **MaxPooling2D**: 0 parameters
- **Conv2D (64 filters)**:

$$(3 \times 3 \times 32 + 1) \times 64 = (288 + 1) \times 64 = 18,496 \text{ parameters}$$

- **MaxPooling2D**: 0 parameters
- **Conv2D (128 filters)**:

$$(3 \times 3 \times 64 + 1) \times 128 = (576 + 1) \times 128 = 73,856 \text{ parameters}$$

- **MaxPooling2D**: 0 parameters
- **Flatten Layer**: 0 parameters
- **Dense (128 units, input 2048)**:

$$(2048 + 1) \times 128 = 262,272 \text{ parameters}$$

- **Dense (64 units)**:

$$(128 + 1) \times 64 = 8,256 \text{ parameters}$$

- **Dense (10 units)**:

$$(64 + 1) \times 10 = 650 \text{ parameters}$$

Total Parameters for Original Model: 364,426

Modified Model

- **Conv2D (32 filters, input shape (32, 32, 3))**:

$$(3 \times 3 \times 3 + 1) \times 32 = (27 + 1) \times 32 = 896 \text{ parameters}$$

- **MaxPooling2D**: 0 parameters
- **Conv2D (64 filters)**:

$$(3 \times 3 \times 32 + 1) \times 64 = (288 + 1) \times 64 = 18,496 \text{ parameters}$$

- **MaxPooling2D**: 0 parameters
- **Conv2D (128 filters)**:
$$(3 \times 3 \times 64 + 1) \times 128 = (576 + 1) \times 128 = 73,856 \text{ parameters}$$
- **MaxPooling2D**: 0 parameters
- **Conv2D (256 filters)**:
$$(3 \times 3 \times 128 + 1) \times 256 = (1152 + 1) \times 256 = 295,168 \text{ parameters}$$
- **MaxPooling2D**: 0 parameters
- **Flatten Layer**: 0 parameters
- **Dense (128 units, input 1024)**:
$$(1024 + 1) \times 128 = 131,200 \text{ parameters}$$
- **Dense (64 units)**:
$$(128 + 1) \times 64 = 8,256 \text{ parameters}$$
- **Dense (10 units)**:
$$(64 + 1) \times 10 = 650 \text{ parameters}$$

Total Parameters for Modified Model: 528,522

The original model has a total of 364,426 trainable parameters. These parameters are distributed across three convolutional blocks and three dense layers. The Conv2D layers contribute the most to the parameter count, followed by the dense layers.

The modified model, which includes an additional convolutional block with 256 filters, increases the total number of trainable parameters to 528,522. This results in an increase of 164,096 parameters compared to the original model.

The addition of the extra convolutional block significantly increases the model's capacity to learn complex features:

- **Original Model:** 364,426 parameters
- **Modified Model:** 528,522 parameters
- **Increase:** 164,096 additional parameters

This increase allows the model to potentially achieve better performance, especially on more complex datasets, but it also requires more computational resources and may risk overfitting if not properly managed.

Problem 2 (Batch Normalization)

For Problem 1, assess the impact of batch normalization on learning speed and accuracy.

Solution to Problem 2

To assess the impact of batch normalization, we added batchnormalization layer after each Conv2D or Dense layer for the original model in problem 1. Then, we trained the original model and batch-normalized original model in a batch size of 64 and 20 epochs, and finally get the result of the training speed and accuracy, which are shown below:

Training and Validation Comparison

- **Learning Speed:** From the plot below, the model with batch normalization shows quicker initial convergence, achieving higher training accuracy in the early epochs compared to the original model. This indicates that batch normalization accelerates learning by stabilizing the optimization process.
- **Accuracy:** The original model achieved a training accuracy of 0.9639 and a validation accuracy of 0.7320. In comparison, the model with batch normalization achieved a slightly higher training accuracy of 0.9749 and a validation accuracy of 0.7419. This indicates that batch normalization helps stabilize and improve generalization, as observed by the slightly better training and validation performance.

A figure showing the comparison between the original model and batch-normalized original model is as below:

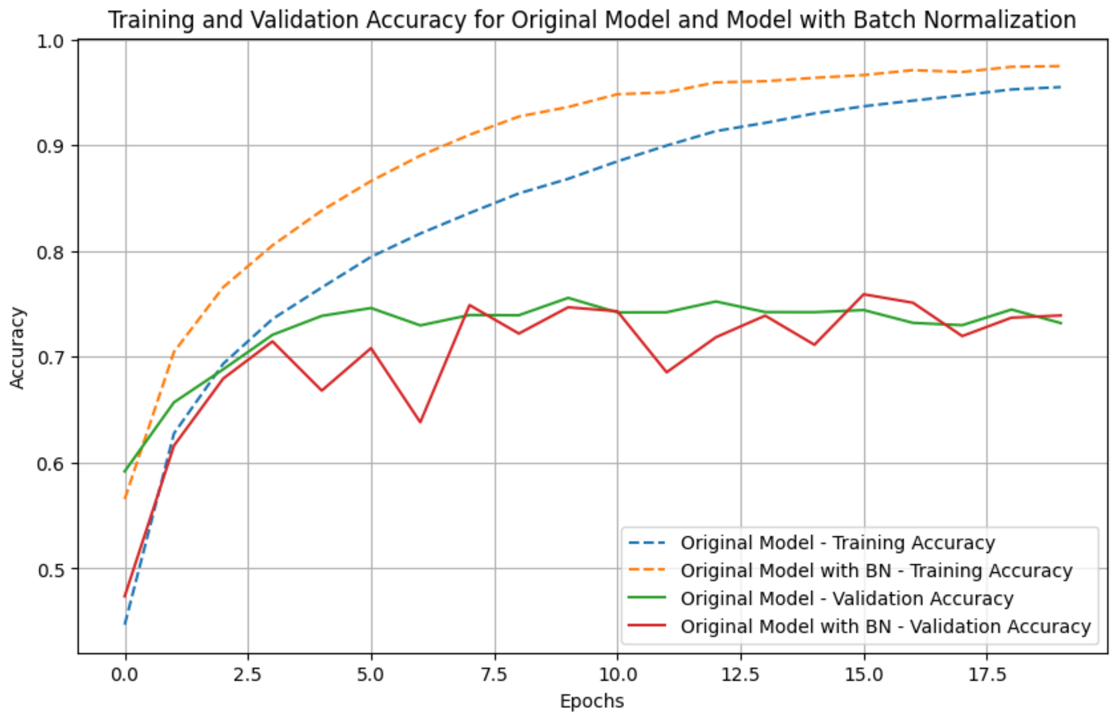


Figure 4: Accuracy Comparison

In summary, the use of batch normalization in the model enhances learning speed and generalization. The training and the validation accuracy benefits, indicating improved model robustness. This highlights the effectiveness of batch normalization in promoting faster learning and better overall performance on validation data.