CNNs

Matei Vlad Cristian

AII

Task 1

What should be noted is that I still have an error in the convolution operation. I fail 2 tests because of it.

```
[14]: convTests.test_dilation()

1
2
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Cell In[14], line 1
----> 1 convTests.test_dilation()

Cell In[1], line 177, in ConvTests.test_dilation(self)
    167 output = conv.forward(random_input)
    168 test_output = F.conv2d(
    169     input=random_input,
    170     weight=conv.weight,
   (...)
    174     padding=0
    175 )
--> 177 self.assertTrue(torch.allclose(output, test_output, atol=ATOL))

File ~\AppData\Local\Programs\Python\Python312\Lib\unittest\case.py:727, in TestCase.assertTrue(self, expr, msg)
    725 if not expr:
    726     msg = self._formatMessage(msg, "%s is not true" % safe_repr(expr))
--> 727     raise self.failureException(msg)

AssertionError: False is not true
```

```
.7]: convTests.test_grouping()

---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[17], line 1
----> 1 convTests.test_grouping()

Cell In[1], line 186, in ConvTests.test_grouping(self)
    183 conv = groups_size_stub_constructor(num_groups=group)
    184 random_input = torch.randn(4, 128, 64, 64)
--> 186 output = conv.forward(random_input)
    187 test_output = F.conv2d(
    188     input=random_input,
    189     weight=conv.weight,
   (...)
    193     padding=0
    194 )
    196 self.assertTrue(torch.allclose(output, test_output, atol=ATOL))

File D:\Sc\CNN_HW\code\conv.py:54, in MyConvStub.forward(self, x)
     52         for h_out in range(output_height):
     53             for w_out in range(output_width):
---> 54                 output[b, c_out, h_out, w_out] = self._conv_forward(h_out, w_out, c_out, b, x)
     56 return output

File D:\Sc\CNN_HW\code\conv.py:70, in MyConvStub._conv_forward(self, h_out, w_out, c_out, b, input_data)
     67 field = input_data[b, :, h_start:h_end:self.dilation, w_start:w_end:self.dilation]
     68 weight_channel = self.weight[c_out, :, :, :]
---> 70 field = field * weight_channel
     71 output = field.sum()
     73 if self.bias is not None:

RuntimeError: The size of tensor a (128) must match the size of tensor b (32) at non-singleton dimension 0
```

The other tests pass successfully.

Task 2

To be honest, when I saw that I had to choose an architecture with a maximum of 5 layers, I instantly thought of LeNet. Therefore, the implementation of LeNet for a 3-channel image is carried out in this project. However, a few explanations:

i) I believe that choosing a kernel size of 5 is appropriate for the level of complexity of such an architecture

ii) The first convolution operation has an input of only 3 by 6, as it expects 3 channels (an RGB image), and the number of 6 filters could be suitable for extracting basic features.

iii) The second convolution operation has an input of 6, according to the output from the first convolution operation. I kept the same kernel size but increased the number of filters, as the complexity also increases at this level (higher-level features).

iv) I used Batch Normalization between the layers because it can definitely help the model remain more stable.

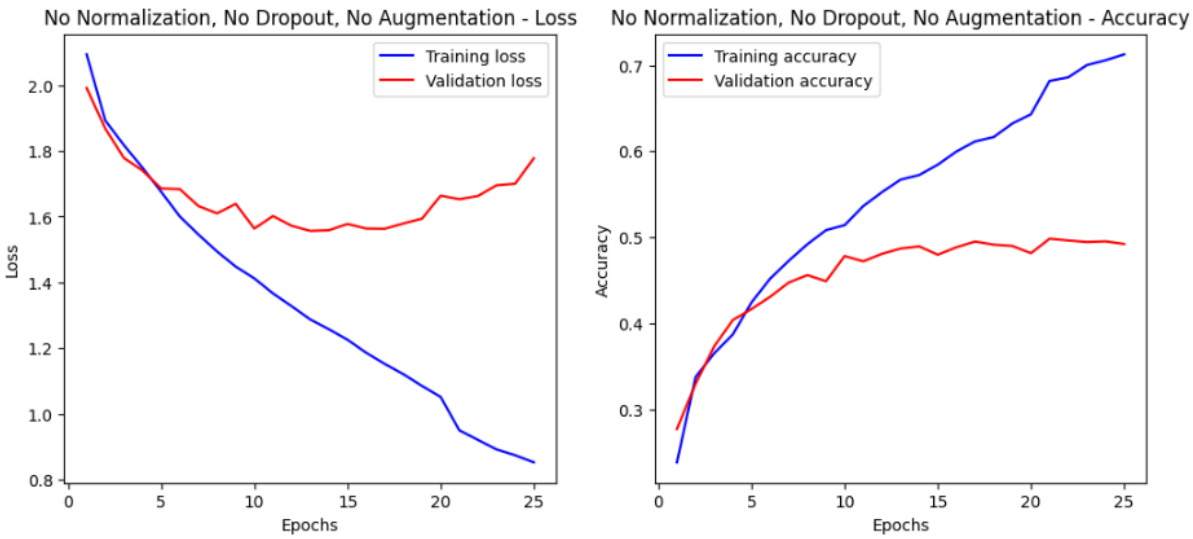
v) The dropout step was required in the specifications 😊, but it definitely helps prevent overfitting as well.

vi) The AvgPool step also helps prevent overfitting and extracts the most important features learned from the image.

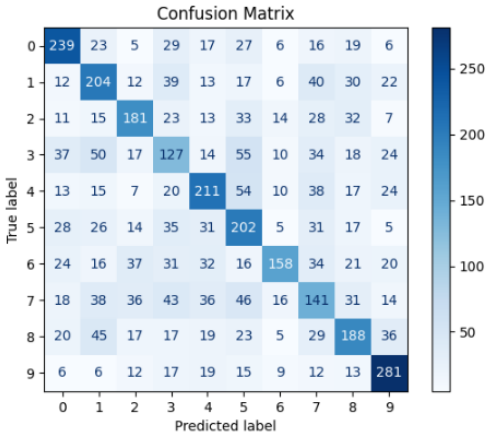
vii) And of course, as is common today, I used ReLU as the activation function.

Experiments explanation:

No Batch Normalization, No Dropout Regularization, No DataAugmentation

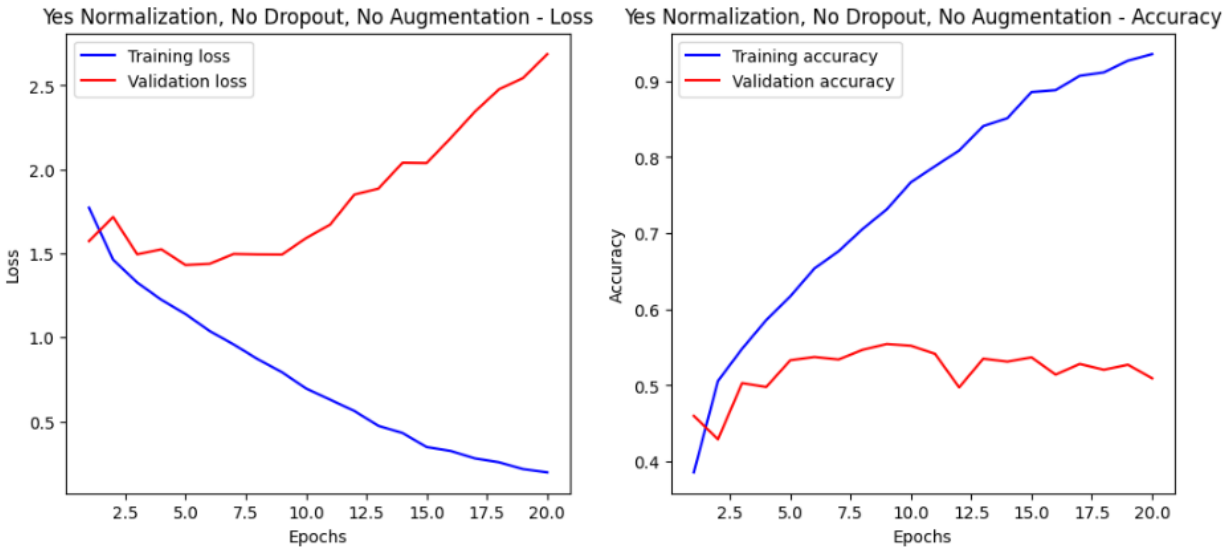


Training took place over 25 epochs, with the StepLR scheduler applied after 20 epochs with a gamma of 0.5. As can be seen, overfitting occurs, starting around epochs 12-15. Additionally, the validation result is low, approximately 0.49. Certainly, the aforementioned methods (BN, DA, DR) are needed for stability.

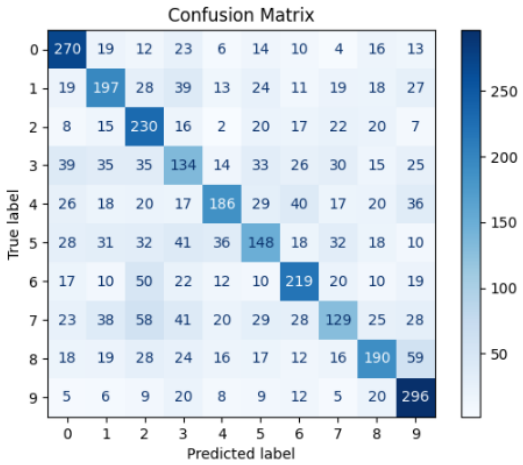


In the confusion matrix, there are a lot of mistakes. Notably, the pairs of True_Label <-> Predicted_Label: 1-7, 3-1, 3-5. Generally, the confusions involve 7-1, 7-2, 7-3, 7-4, 7-5, 7-8. Class 7 seems to be easily confused. Class 7 is “garbage truck.” It is likely to be confused with others because it is a more complex vehicle and can be easily mistaken for a house, a box, a church, or simply a backyard where a dog might have a house or a car in the background. Clearly, the model is not performing very well either.

Yes Batch Normalization, No DR, No DA

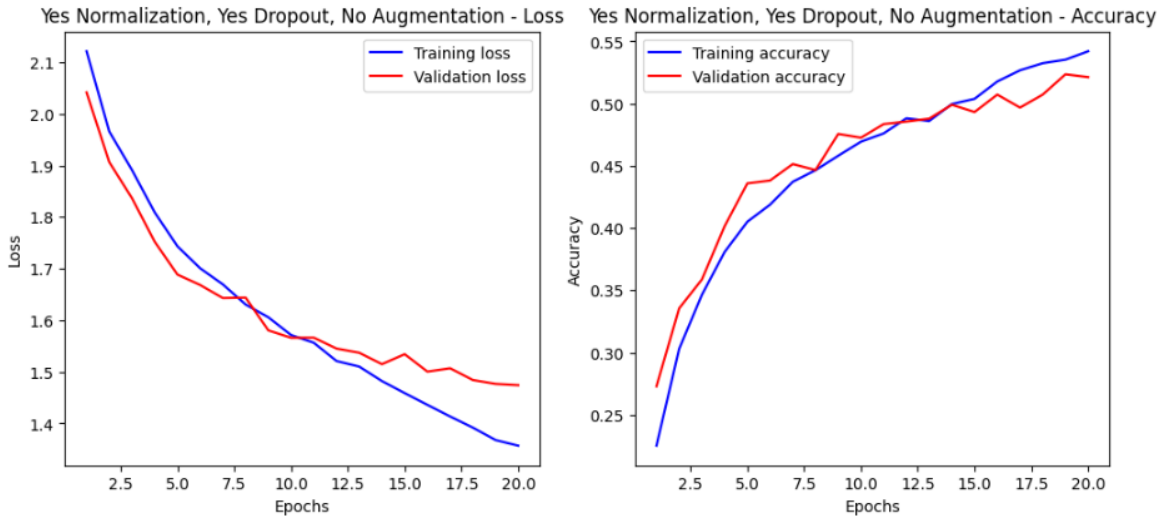


The training took place with Batch Normalization for approximately 20 epochs. As can be seen, there is still a lot of overfitting; it seems that just using Batch Normalization is not enough. Additionally, the learning rate may have been too high. However, overall, the final score is slightly better, or at least shows potential, if a lower learning rate had been used. (I could have changed it—I've mentioned wanting diversity to explain different phenomena and demonstrate improvements along the way + explanations.) (It took 10 minutes to retrain.)

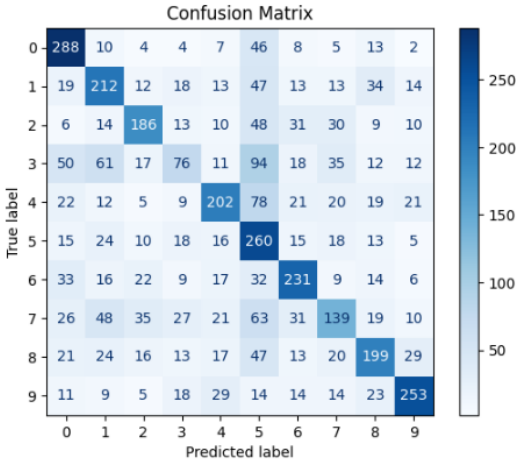


Regarding class 7, it seems that confusion is more common. The results are not good at all.

1. **No BN, Yes Dropout Regularization, No DA**

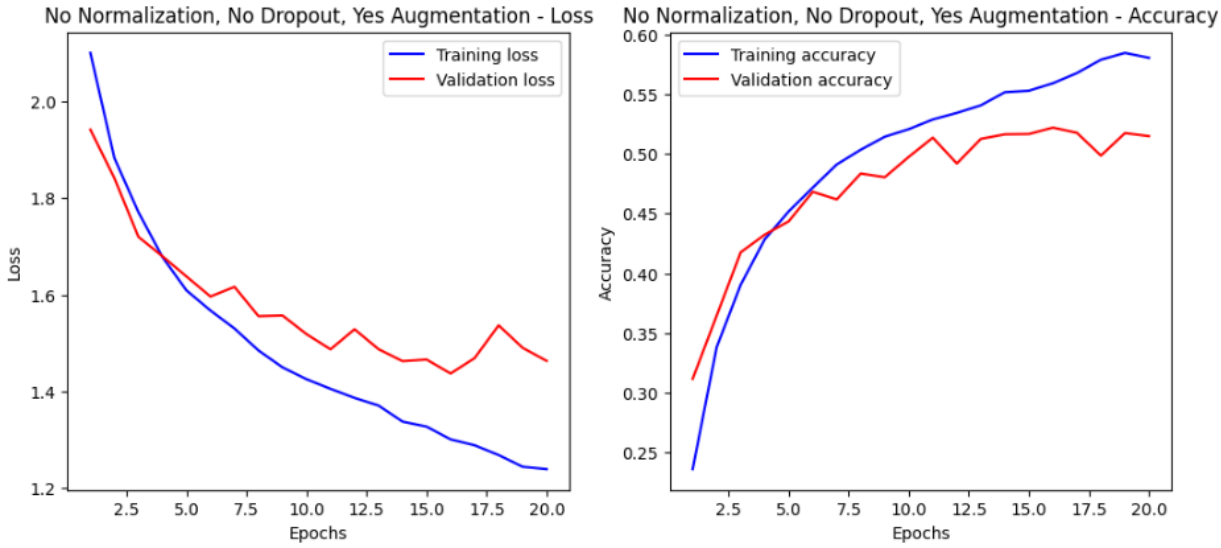


Perhaps the best results from this category, where each improvement is tested one at a time. The loss and accuracy curves are very "nice"; the model converges well. So far, it proves to be the best method. (The experimental setup remains unchanged.)



There isn't much to say; there are still issues. Classes 3 and 7 have many problems. However, there is also confidence regarding classes 0 and 9.

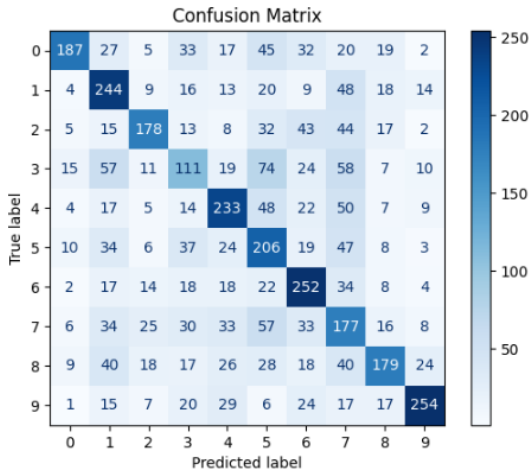
2. No BN, No DR, Yes Data Augmentation



As can be seen, the results are much better than with Batch Normalization. The training took place over 20 epochs, identical to the other scenarios. The learning rate remained at 0.001, but the results are definitely better. Yes, they are not perfect, but overfitting is no longer noticeable as it was in the first case.

As for the augmentation methods, I used:

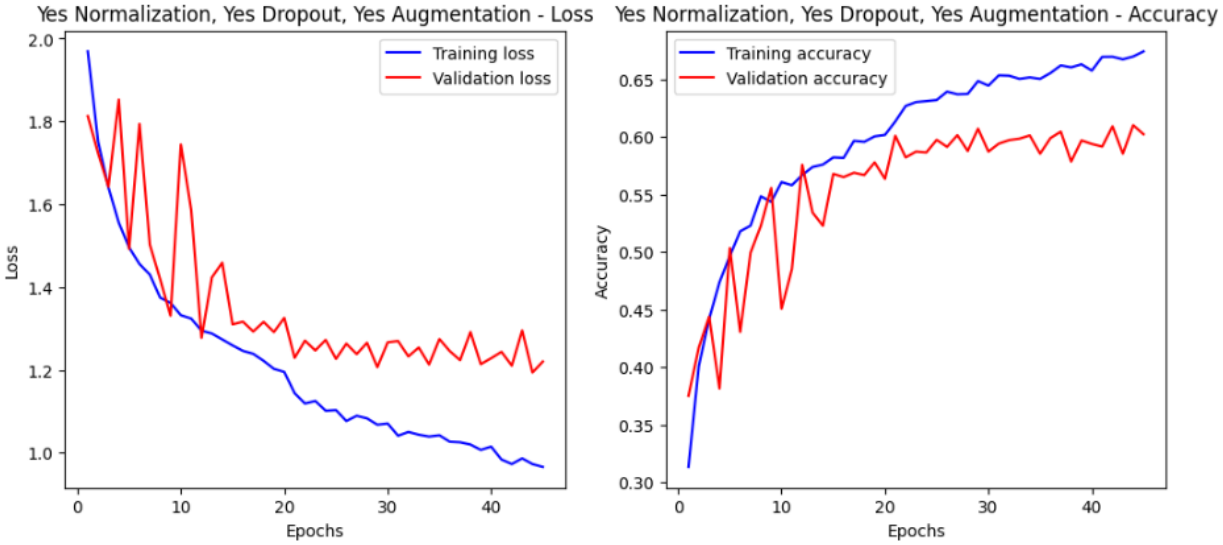
- **RandomHorizontalFlip()** – for robustness, better generalization, and preventing overfitting. I believe this is a basic method that "flips" the patterns for more accurate learning.
- **ColorJitter()** ... which alters brightness, contrast, saturation, and other hues. Similarly, this is aimed at improving robustness. Some objects may not appear frequently in the training set in certain combinations of colors and contrasts, but they can still exist in reality as objects in such environments. Thus, the model becomes more robust.



Regarding the confusions, there are significant confusions between class 3 and class 5. Additionally, there are confusions between class 5 and class 7. However, other confusions, such as between classes 7-0, 7-8, 7-9, and to some extent 7-1, are improving. So, the model is making progress. It is changing its weights, and it is clear that, with the help of Gradient Descent, it reaches an area that was still unexplored in the initial experiments.

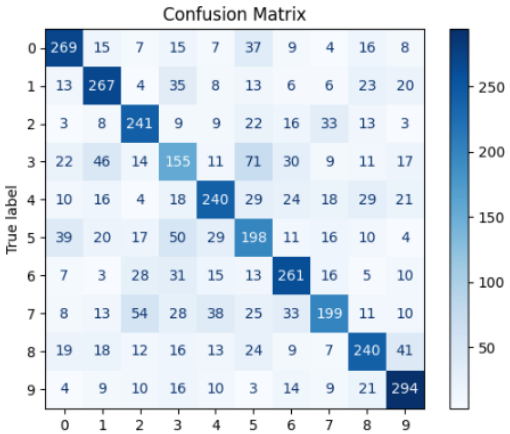
3. Yes BN, Yes DR, Yes DA

As expected, all these combined methods, along with an increased number of epochs (45) and a better-defined scheduler, have led to improved results.



I would like to point out that the scheduler could have been used more wisely. Nonetheless, the results converge well, and we have achieved 60% accuracy.

```
Epoch [41/45], Training Loss: 0.9838, Training Accuracy: 0.6696, Validation Loss: 1.2438, Validation Accuracy: 0.5916
Epoch [42/45], Training Loss: 0.9730, Training Accuracy: 0.6697, Validation Loss: 1.2105, Validation Accuracy: 0.6092
Epoch [43/45], Training Loss: 0.9866, Training Accuracy: 0.6674, Validation Loss: 1.2954, Validation Accuracy: 0.5855
Epoch [44/45], Training Loss: 0.9728, Training Accuracy: 0.6699, Validation Loss: 1.1940, Validation Accuracy: 0.6102
Epoch [45/45], Training Loss: 0.9659, Training Accuracy: 0.6743, Validation Loss: 1.2204, Validation Accuracy: 0.6023
```



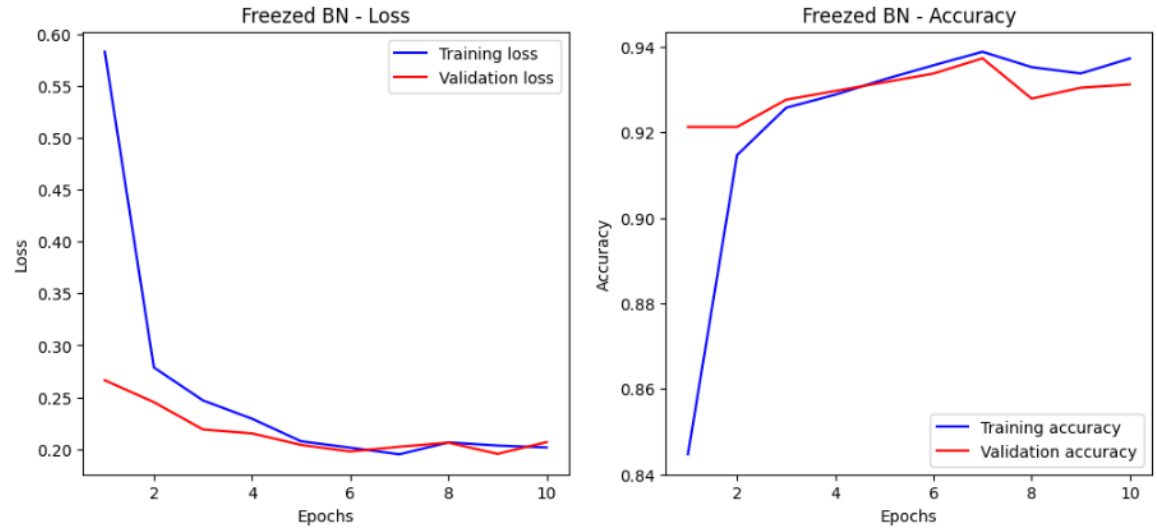
It can be observed that the model resolves most of its confusions. However, there are still a few "problematic" classes remaining, such as classes 3, 5, and 7.

Task 3

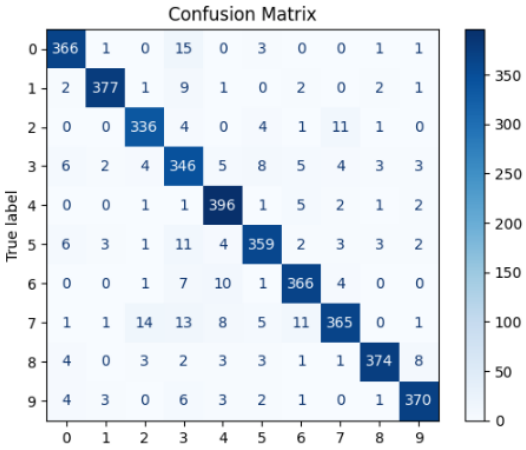
When I froze everything, including Batch Normalization, I obtained these results:

```
Epoch 10/10
----------
Training Loss: 0.2016, Training Accuracy: 0.9373
Validation Loss: 0.2067, Validation Accuracy: 0.9312
```

```
plot_curves(train_losses, val_losses, train_accs, val_accs, 'Freezed BN')
```



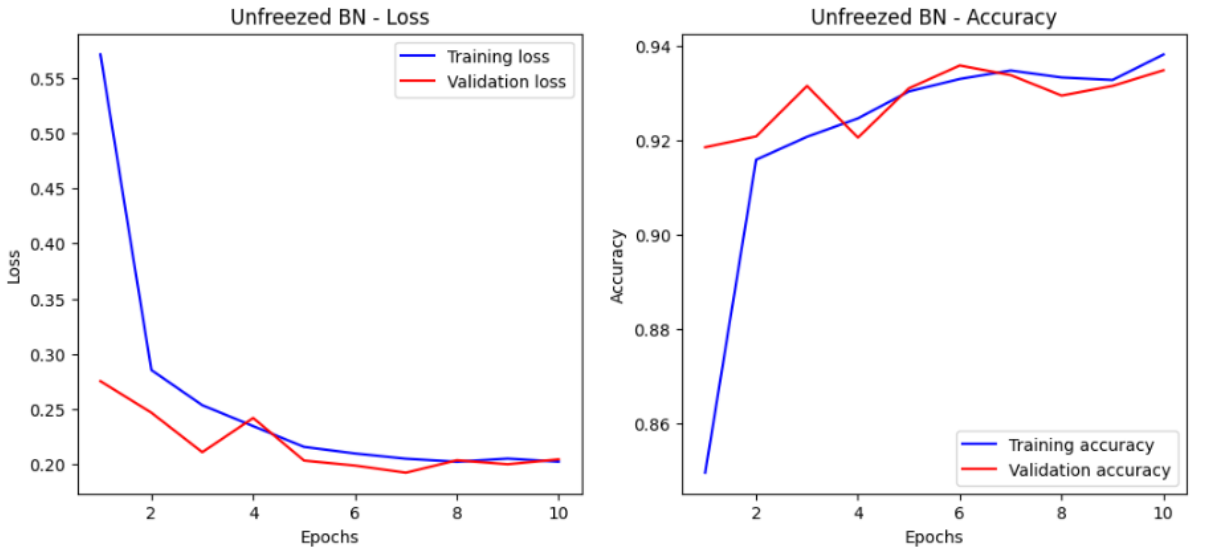
It can be observed that there is an acceptable convergence curve, perhaps even a tendency towards maximum convergence. The results are good; it is challenging to achieve perfection in every case, so it's acceptable to have some errors, even though I used a pre-trained model on ResNet18-ImageNet. This model has already learned very complex features, making this task easier. Training just the last classification layer is sufficient for the model to be adapted. Additionally, a very large number of training epochs is no longer necessary. With just 10 epochs, or even fewer, the results converge very well and quickly.



Regarding the confusion matrix, classes 5 and 7, which previously had many issues, now perform significantly better. The results are much improved, with only a few natural confusions; however, the final accuracy is 93% on the validation dataset.

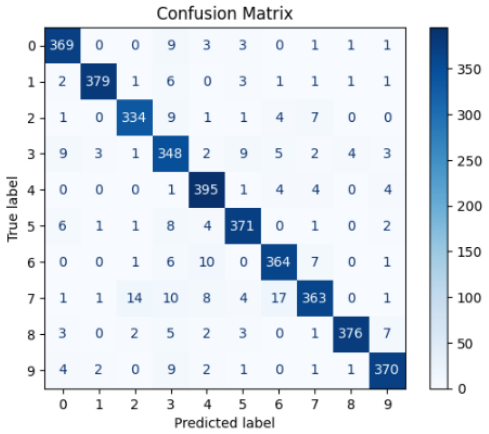
When the Batch Normalization (BN) layers were also trained, the training certainly took a bit longer due to the increased number of weights to adjust. The loss curve is influenced by both the scheduler and the learning rate. Nonetheless, a nuanced improvement in the model can be observed.

```
plot_curves(train_losses, val_losses, train_accs, val_accs, 'Unfreezed BN')
```




```
Epoch 10/10
----------
Training Loss: 0.2026, Training Accuracy: 0.9381
Validation Loss: 0.2047, Validation Accuracy: 0.9348
```

It seems that the model is slightly more stable in its convergence. There are more "corners" in the lines of the graph, indicating that the model is making sharper changes more frequently, but ultimately, the trajectory appears better than in the previous case. Overall, the process takes longer, but it seems to help, albeit just a little.



Minimal differences regarding the confusion matrix.