COMP90086 Computer Vision

Assignment 2– Semester 2, 2024

Name: Luxi Bai

Number: 1527822

Email: LUXIB@student.unimelb.edu.au

Question 1: Baseline Neural Network

After running the baseline neural network in the notebook, I obtained the accuracy curve (Figure 1) and the loss curve (Figure 2) for the model. The blue line represents the training set performance, while the orange line shows the validation set performance.

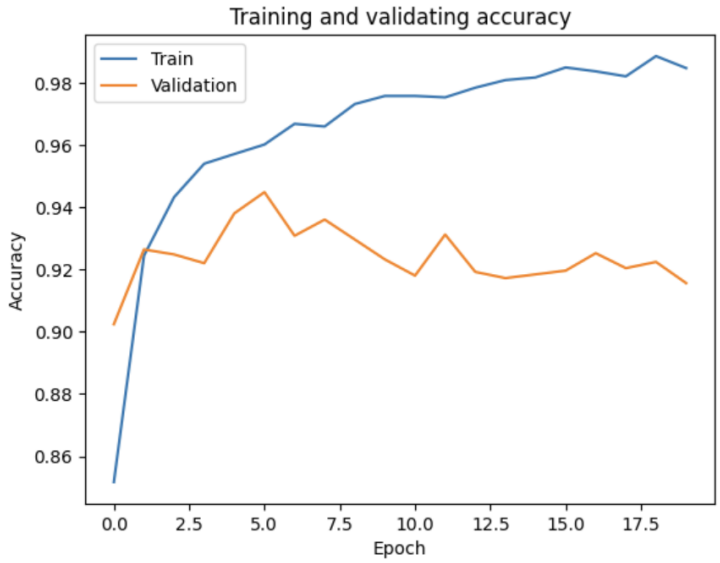


Figure 1 Training and validating accuracy

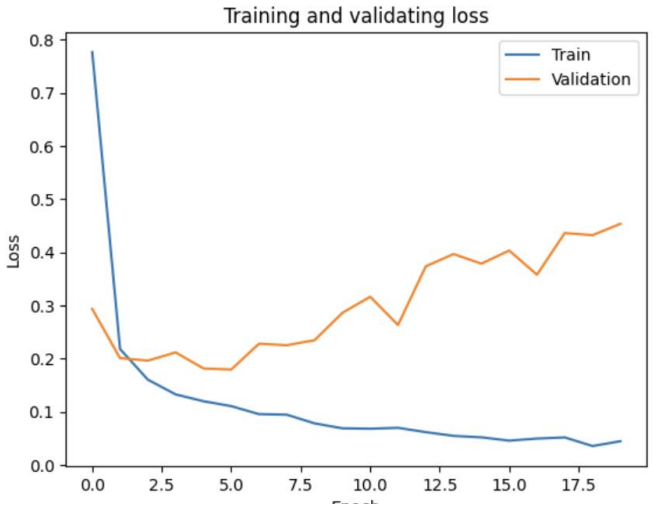


Figure 2 Training and validating loss

From the accuracy curve, we can see that the training accuracy stabilizes at over 98% after 10 epochs, indicating that the model performs very well on the training data. However, the validation accuracy initially rises quickly to around 93% but then starts to decline, showing significant fluctuations. Eventually, it stabilizes below 92%. This suggests that the baseline model is likely overfitting, meaning the neural network has learned too many specific features from the training set, resulting in poor generalization to unseen data. The loss curve further confirms this issue. While the training loss steadily decreases, the validation loss drops from 0.3 to 0.2 in the early stages, but then begins to increase with substantial oscillations, reaching nearly 0.5 by the end. This indicates that the model struggles to generalize well to the validation data, as it has overfit to the training examples and fails to capture certain features in the test set.

Despite the overfitting, it's worth noting from the accuracy curve that this task appears relatively simple. Even with overfitting, the model maintains a validation accuracy above 92%, suggesting that distinguishing whether two image patches correspond is not too difficult.

Regarding the imbalance in the validation set, where there are more "bad" examples (non-matching patches) than "good" examples (matching patches), this reflects a realistic scenario where, in actual matching tasks, keypoints from different viewpoints or scenes are much more likely to not match than to match. The purpose of the model is to address this issue effectively, and therefore, having a validation set with a higher proportion of negative cases (bad : good= 4:1) is reasonable. By increasing the proportion of negative samples, we can better assess the model's ability to distinguish non-matching features and evaluate its performance in real-world applications.

Whether this kind of imbalance needs to be reflected in the training set also needs to be judged according to the actual situation. If we hope that the model can better distinguish the cases of mismatch, we can appropriately increase the negative samples. If there is no difference in the importance given to positive and negative samples, a 1:1 situation can be set. The ratio given in this question is 1:1, which indicates to some extent that the characteristics of both situations are expected to be fully learned during training.

Question 2: Regularizing your Neural Network

From the first analysis, it is clear that the baseline model suffers from overfitting. To address this issue, I experimented with regularization techniques to observe their effects on the model. After the Flatten() layer, I added a Dropout() layer and tested different dropout rates. I examined models with dropout values of 0.5 (Figure 3), 0.8 (Figure 4), and 0.9 (Figure 5), and generated the accuracy and loss graphs for each model.

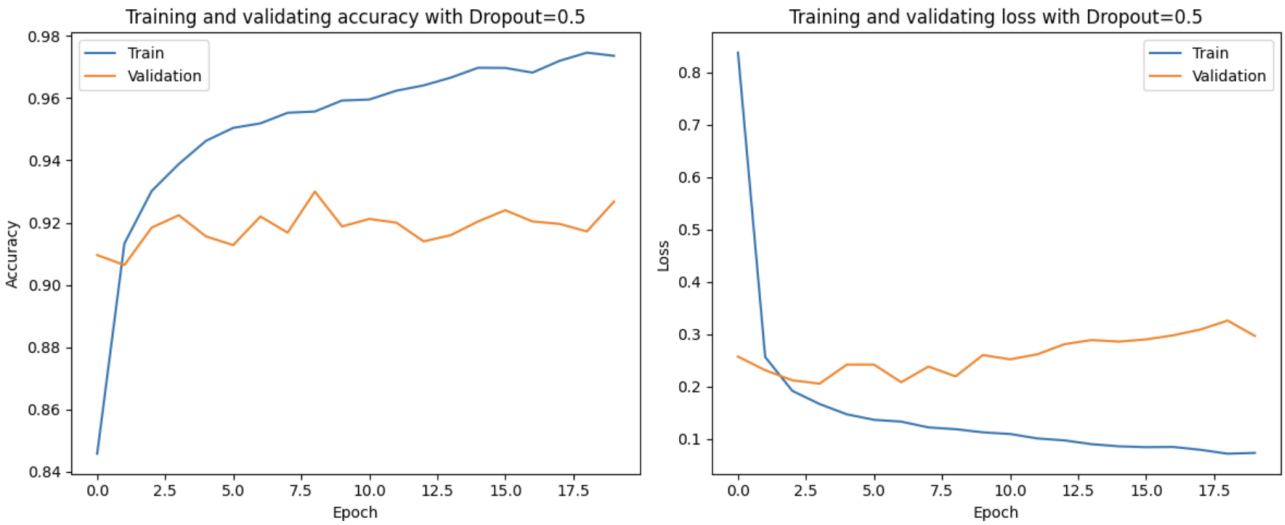


Figure 3 Accuracy and loss with Dropout=0.5

For the model with a dropout rate of 0.5, the training accuracy (Figure 3, left) stabilized around 0.96, while the validation accuracy hovered around 0.92 (Figure 3, right), showing significant fluctuations. This indicates that the model still suffers from overfitting and lacks stability. This is also evident from the loss curves, where the training loss drops below 0.1 after 10 epochs, but the validation loss remains above 0.3. Therefore, further adjustments to the

dropout rate are necessary.

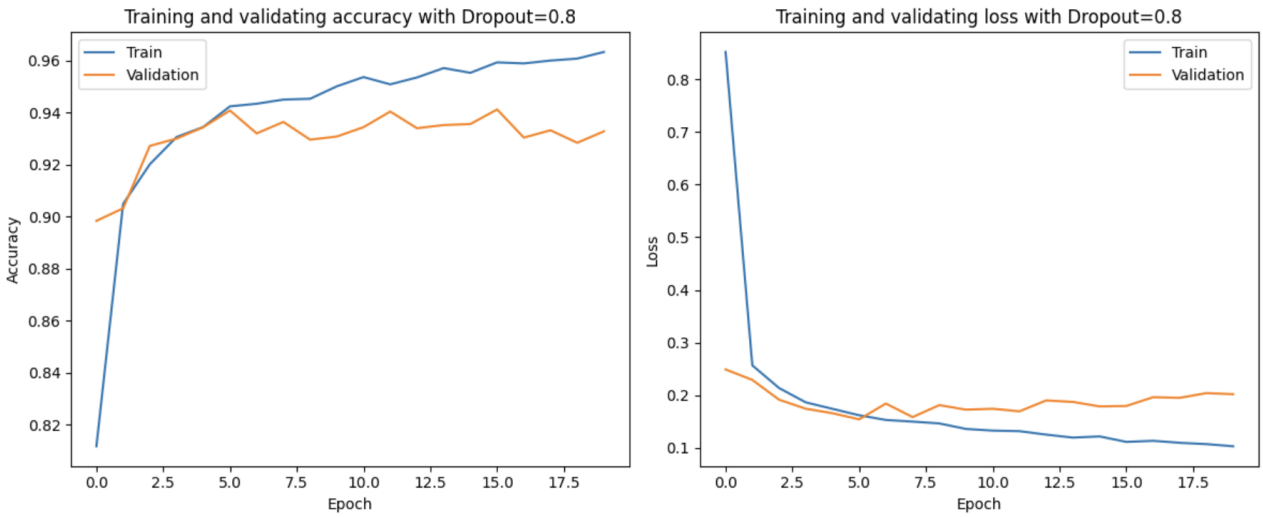


Figure 4 Accuracy and loss with Dropout=0.8

Next, I tested a dropout rate of 0.8. In this model, the accuracy (Figure 4, left) shows that both training and validation accuracies are much closer, both around 0.94. Although the validation accuracy slightly drops after 17.5 epochs while the training accuracy reaches 0.96, the overall results are quite satisfactory. Additionally, the loss for both the training and validation sets remains around 0.15, with much less fluctuation, indicating a relatively stable and well-performing model.

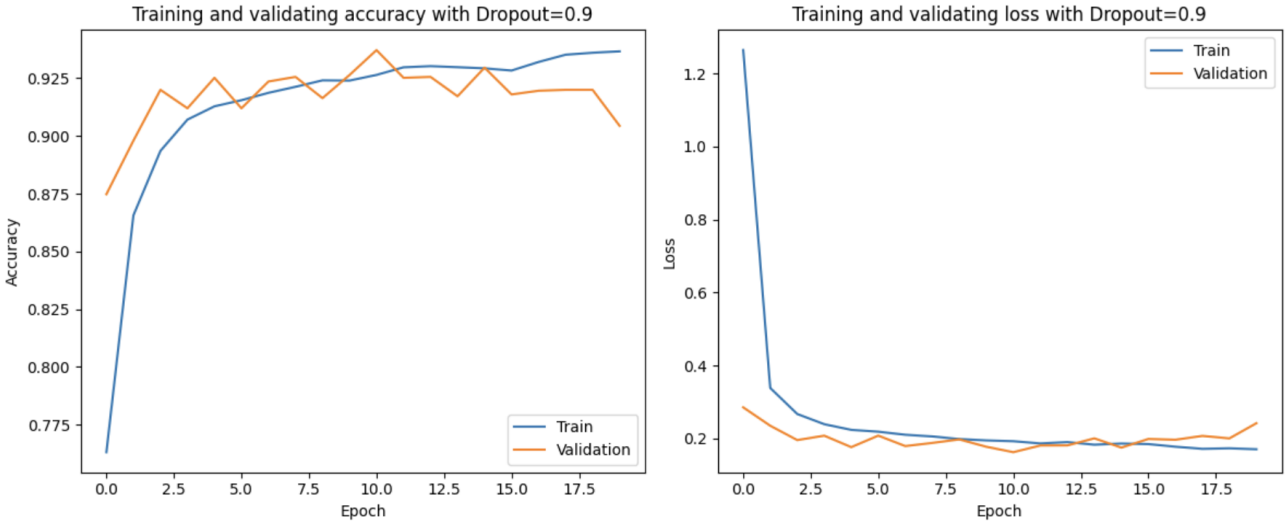


Figure 5 Accuracy and loss with Dropout=0.9

Since the previous model performed well, I increased the dropout rate to 0.9 to see if further improvement was possible. The results in Figure 5 show that the training and validation accuracies are nearly identical, both fluctuating around 0.92, and the loss stabilizes around 0.2. However, despite addressing the overfitting issue, the overall performance declined. The accuracy dropped from around 0.94 in the 0.8 dropout model to 0.92, and the loss increased from 0.15 to 0.2. This suggests that while overfitting has been effectively mitigated, the overall model performance has suffered.

Therefore, among the three models, I believe the model with a dropout rate of 0.8 is the best. In this model, the accuracy and loss for both the training and validation sets are well-balanced, and the overfitting issue is resolved without compromising overall performance.

Question 3: Convolutional Neural Network

The neural network in the second question still has certain limitations. Therefore, in the third task, I attempted to design a Convolutional Neural Network (CNN) model to address these issues. In this task, I experimented with ten different configurations, adjusting parameters such as kernel size, filter numbers, strides, MaxPooling kernel size, MaxPooling strides, and different fully connected layer structures. The results of the ten different training runs are summarized in the table below:

Setting number	Convolution layer structure	Kernel size	Filter size	Strides	Drop out	MaxPooling kernel size	MaxPooling strides	Fully connected layer structures	Training Acc	Training loss	Validation Acc	Validation loss
1	1 layer	3*3	32+64	1	0.2	\	\	16+relu 2+softmax	0.984	0.059	0.735	1.645
2	1 layer	3*3	64+128	1	0.2	\	\	16+relu 2+softmax	0.501	0.693	0.800	0.692
3	1 layer	5*5	32+64	1	0.2	\	\	16+relu 2+softmax	0.990	0.035	0.872	0.771
4	2 layer	3*3*3	32+64	1	0.2	1*2*2	\	16+relu 2+softmax	0.997	0.008	0.987	0.097
5	2 layer	3*3	32+64	1	0.2	3*3	1	16+relu 2+softmax	0.971	0.088	0.830	0.772
6	2 layer	3*3	32+64	1	0.1	3*3	1	16+relu 2+softmax	0.919	0.238	0.658	1.763
7	2 layer	5*5	32+64	1	0.2	5*5	1	16+relu 2+softmax	0.874	0.314	0.758	0.546
8	2 layer	3*3	32+64	2	0.2	3*3	2	16+relu 2+softmax	0.922	0.202	0.918	0.197
9	2 layer	3*3	64+128	1	0.2	3*3	1	16+relu 2+softmax	0.892	0.270	0.802	0.546
10	2 layer	3*3	32+64	1	0.2	5*5	1	32+relu 4+softmax	0.983	0.048	0.902	0.406

Based on the results of running different sets of data, it is evident that various parameter configurations significantly affect the model's performance. Accuracy and loss figures are attached below as Figure 6 to Figure 15.

In the first and second sets, where all parameters were kept constant except for the filter size, the results showed that the 32+64 filter size combination performed far better than 64+128. When the filter size was too large, the accuracy on the validation set exceeded that on the test set, and the loss rate increased significantly. This indicates that an excessively large filter size causes the model to lose important details. After testing a 16+32 filter size, I found that its performance was worse than the 32+64 combination, as it showed signs of underfitting, with lower accuracy and a higher loss rate. This highlights the importance of choosing an appropriate filter size, as sizes that are too large or too small negatively impact the model’s performance.

In the comparison between the first and third sets, where the kernel size was increased from 3x3 to 5x5 while keeping all other parameters constant, the results showed an improvement in accuracy on both the training and test sets, along with a slight decrease in loss. This suggests that a larger kernel size can offer some optimization for the model.

For the first and fifth sets, an additional convolutional layer was added in the fifth set, while all other parameters remained the same. The results showed that adding a second convolutional layer reduced the accuracy gap between the training and test sets, and the loss rate also decreased compared to the single-layer model. This indicates that adding convolutional layers appropriately can enhance model performance.

The comparison between the fourth and fifth sets, where the only difference was the use of Conv3D in the fourth set versus Conv2D in the fifth, demonstrated that the Conv3D model in the fourth set performed exceptionally well. It achieved a 0.997 accuracy on the training set and a 0.987 accuracy on the test set, with a training loss of only 0.008 and a test loss of 0.097. These results are highly ideal in terms of both accuracy and loss.

The fifth and sixth sets, which compared different dropout values while keeping other parameters constant, showed that reducing the dropout rate from 0.2 to 0.1 caused the accuracy on the test set to drop significantly, while the accuracy on the training set remained relatively unchanged. Both the training and test set loss rates increased substantially, indicating that reducing the dropout rate too much leads to severe overfitting.

The comparison between the fifth and seventh sets involved increasing both the kernel size and the max pooling kernel size while keeping all other parameters constant. The results showed a noticeable drop in accuracy on both the training and test sets in the seventh set. While the loss rate increased significantly on the training set, it decreased on the test set, indicating that the model suffered from underfitting. This suggests that excessively increasing the kernel size can cause the model to lose important details, leading to underfitting.

In the comparison between the fifth and eighth sets, where only the max pooling strides were increased from 1 to 2, the results showed that although the accuracy in both the training and test sets was around 0.92, the difference between them was minimal. Additionally, the loss rates were more balanced compared to the fifth set. This indicates that moderately increasing the strides can effectively alleviate overfitting, and the eighth set stood out as one of the most balanced among all the tested configurations.

Finally, the comparison between the fifth and tenth sets involved changing the structure of the fully connected layers. After doubling the structure of the fully connected layers, the tenth set outperformed the fifth in overall performance, although some overfitting remained.

In conclusion, the best-performing model is from the fourth set, which uses two convolutional layers with Conv3D, a kernel size of 3x3, a filter size of 32+64, a dropout rate of 0.2, and a max pooling kernel size of 1x2x2. This model achieves excellent results in terms of both accuracy and loss, with no signs of overfitting or underfitting.

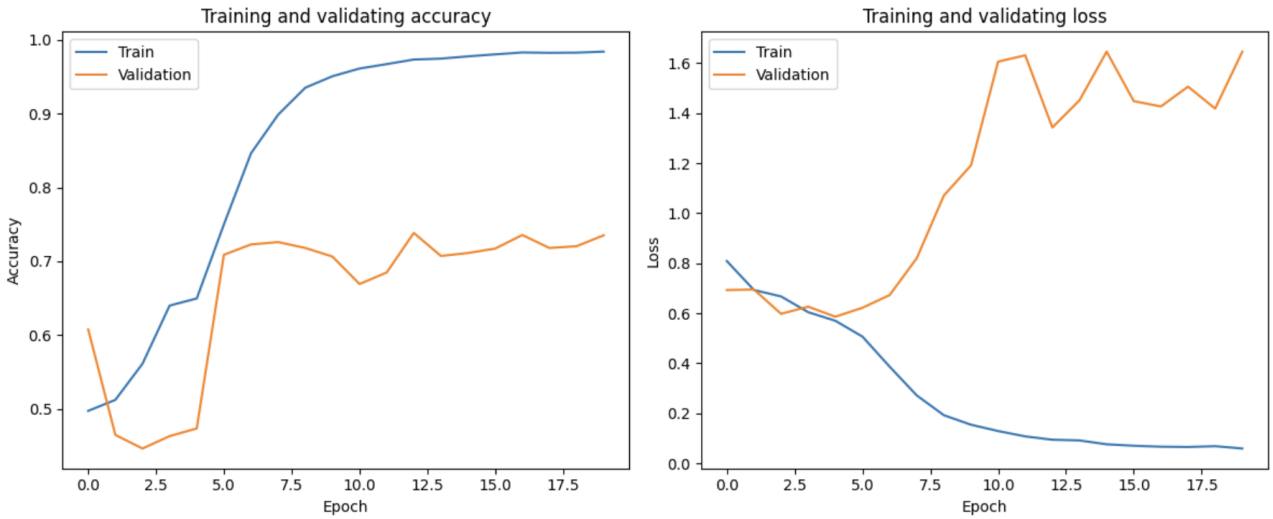


Figure 6 Setting 1

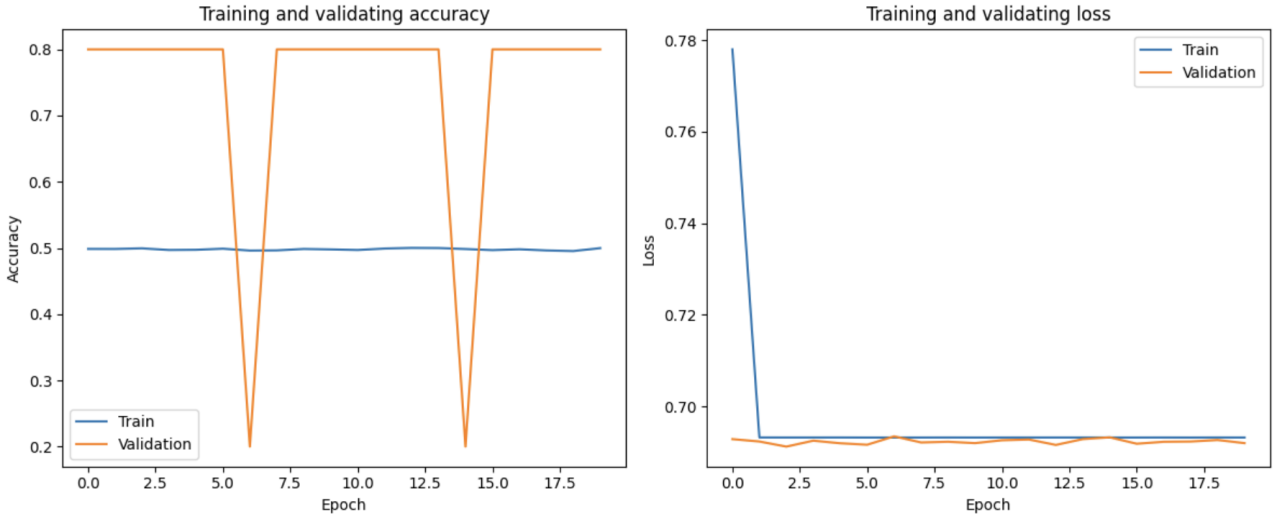


Figure 7 Setting 2

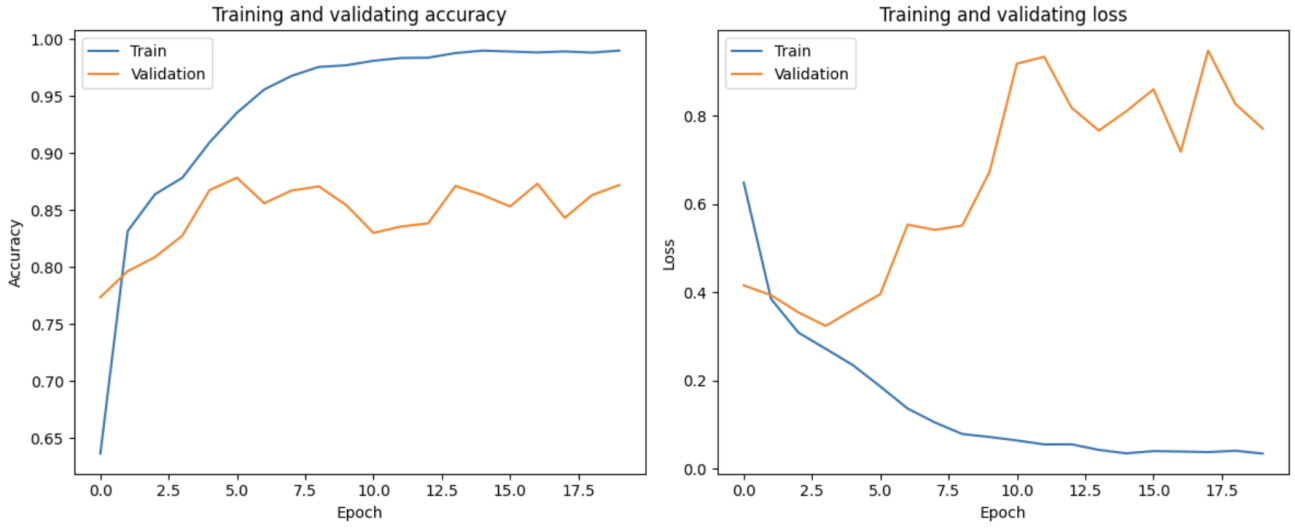


Figure 8 Setting 3

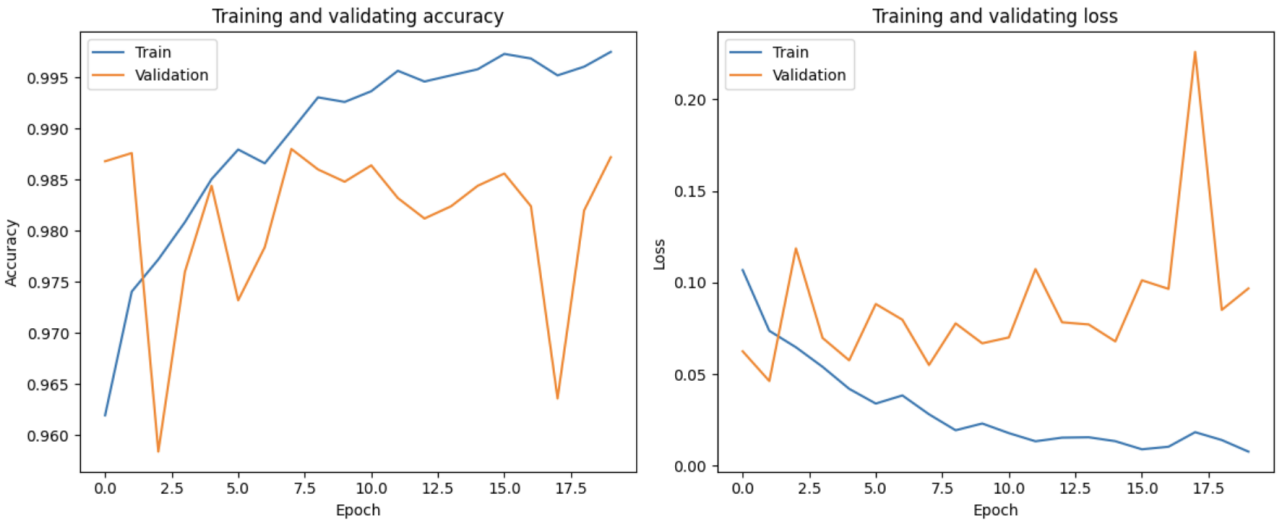


Figure 9 Setting 4

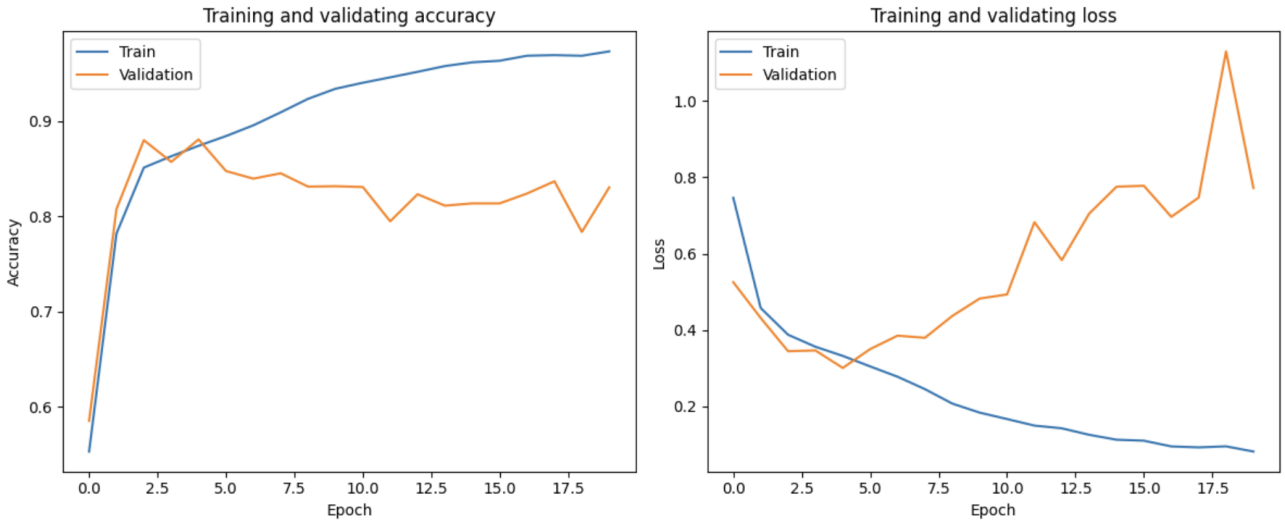


Figure 10 Setting 5

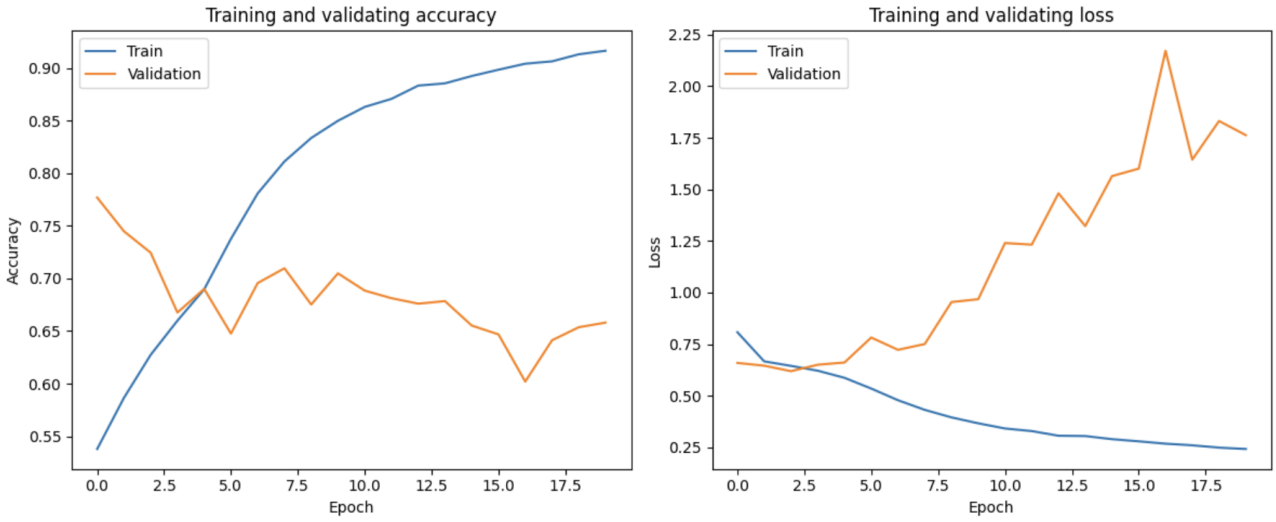


Figure 11 Setting 6

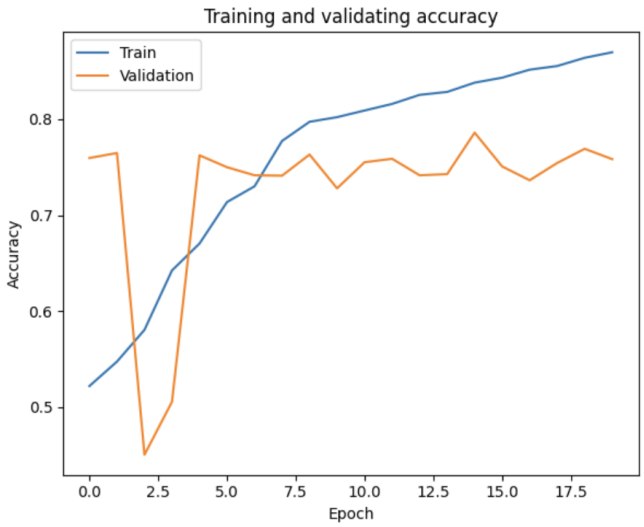
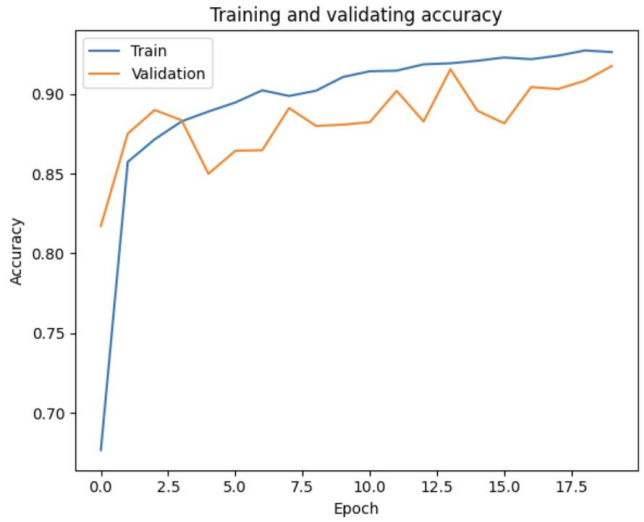


Figure 12 Setting 7



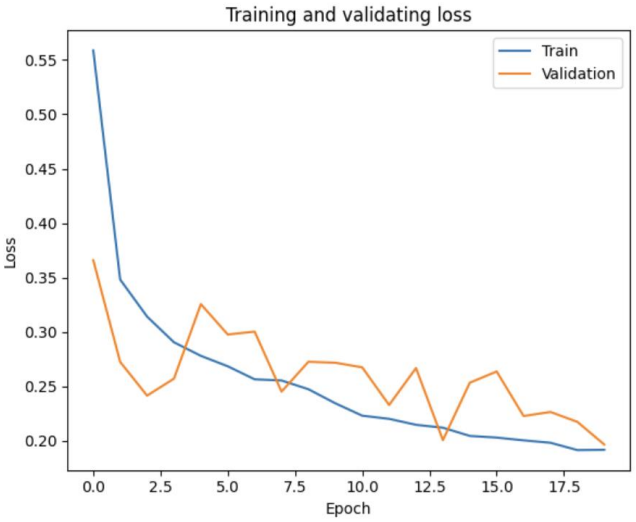


Figure 13 Setting 8

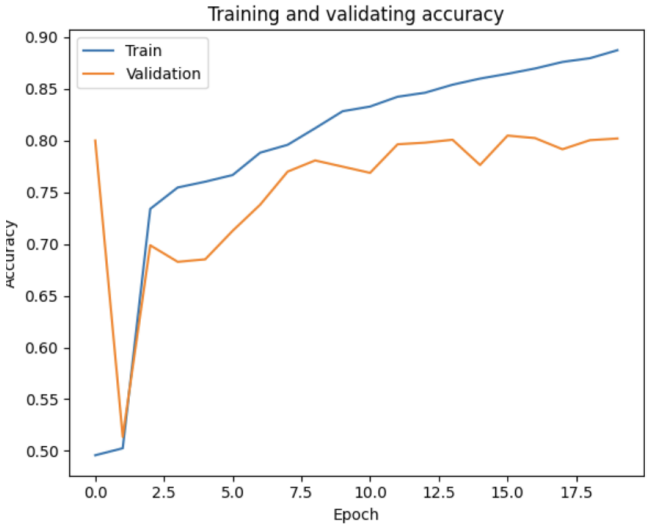


Figure 14 Setting 9

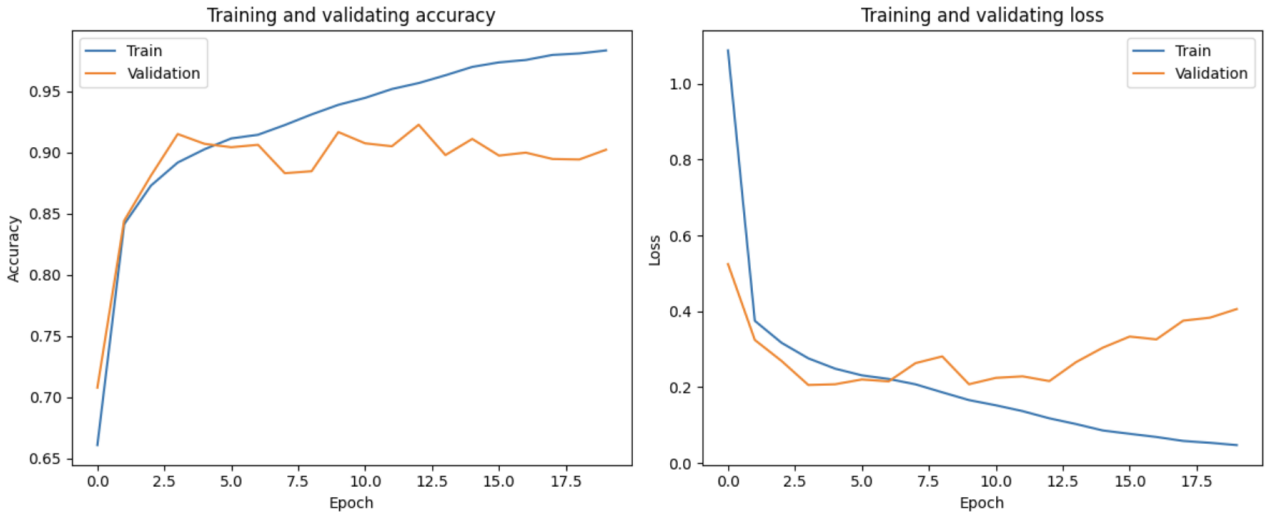


Figure 15 Setting 10