Medical Image Computing CAP5516 Assignment#01

Ashmal Vayani

UCF ID: 5669011

1 Deep Learning-based Pneumonia Detection Using Chest X-Ray Images

Implementation Details:

In this assignment, I used two variants of ResNet18, one trained from scratch and one using pre-trained weights.

Training Code:

The training and implementation code can be accessed **here**.

1.1 Network Architecture:

1.1.1 ResNet18 (Scratch): Training from Scratch

The `ResNet18_Scratch` model initializes a ResNet-18 architecture without pretrained weights (`pretrained=False`). This implies:

- The convolutional layers and fully connected layers are randomly initialized.
- The model must learn all feature representations from the ground up, requiring a larger dataset and longer training time.
- During training, the model updates all parameters via backpropagation.
- A dropout layer (with probability p) is added before the final fully connected (FC) layer to prevent overfitting.
- The final FC layer is modified to output two classes instead of ResNet-18's default 1000-class output.

1.1.2 ResNet18: Transfer Learning with Pre-trained Weights

The `ResNet18` model uses a pretrained ResNet-18 backbone (`weights=models.ResNet18_Weights.DEFAULT`). This means:

- The model is initialized with weights trained on a large-scale dataset (e.g., ImageNet).
- Only the final FC layer is replaced to adapt to a two-class classification problem.
- Since lower layers already contain meaningful features, they can be frozen during initial training.

1.1.3 Comparison of Both Approaches

Table 1 shows the hyper-parameters that were used in Training ResNet18 from Scratch and Pre-trained weights. Table 2 shows the hyper-parameters that are being used in the training of both the Scratch and pre-trained ResNet18 weights. The learning rate is set to 0.001 and I have trained both the variants for 40 epochs.

Feature	Training from Scratch	Using Pretrained Weights
Initialization	Random weights	ImageNet-pretrained weights
Training Time	Longer	Shorter
Data Requirement	Large dataset needed	Smaller dataset sufficient
Feature Learning	Learns from scratch	Uses learned features
Risk of Overfitting	Higher	Lower
Performance	Lower initially	Good from the start

Table 1: Comparison of ResNet-18 training approaches

Hyperparameter	Value
Mode	Scratch/Pretrained
Learning Rate	0.001
Batch Size	32
Number of Epochs	40

Table 2: Hyperparameters for Training ResNet-18 from Scratch

1.2 Training and Validation Loss curves:

The below figure shows the Training, Validation, and Test loss curves on both Scratch (random initialization) and using pre-trained model weights on ResNet18.

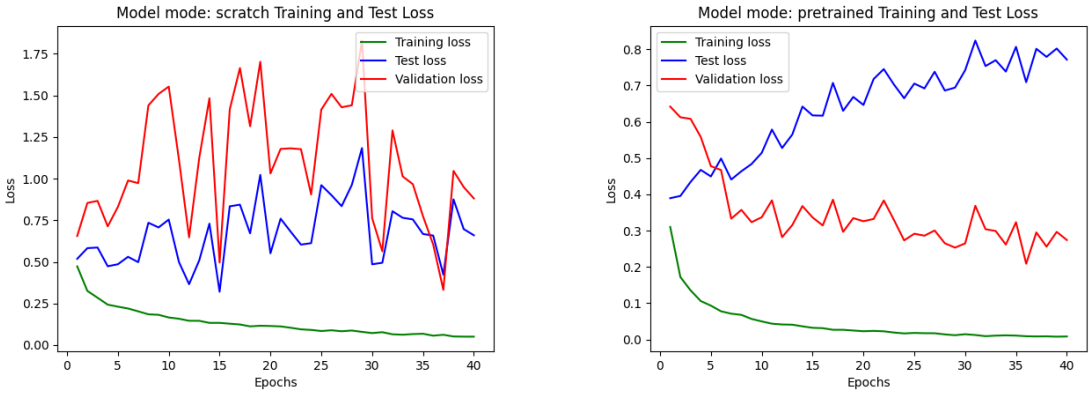

Figure 1: In the pre-trained model (Image 2), the training loss steadily decreases and converges to nearly zero, indicating effective learning on the training data. However, there's a concerning trend where the test loss gradually increases over epochs (from 0.4 to 0.8), while the validation loss remains relatively stable around 0.3-0.4. This divergence between test and validation loss suggests some potential issues with model generalization or dataset distribution differences. In contrast, the model trained from scratch (Image 1) shows much more volatile behavior, particularly in the validation loss which fluctuates dramatically between 0.5 and 1.75. While its training loss similarly decreases to near zero, both test and validation losses show erratic patterns without clear convergence, with high-amplitude oscillations throughout training. The scale of loss values is also notably different - the scratch model experiences much higher loss values overall (up to 1.75) compared to the pre-trained model (maximum around 0.8).

1.3 Training and Validation Accuracy curves:

The above figure shows the Training, Validation, and Test accuracy curves on both Scratch (random initialization) and using pre-trained model weights on ResNet18.

Figure 2: The pre-trained model (Image 2) shows significantly more stable and better performance overall, with training accuracy quickly reaching and maintaining around 100%, while test and validation accuracies stabilize at approximately 82-87%. This indicates good generalization without severe overfitting. In contrast, the model trained from scratch (Image 1) exhibits much more erratic behavior, particularly in its validation accuracy which fluctuates widely between 50-80%. Its training accuracy, while eventually reaching similar high levels as the pre-trained model, takes longer to converge. The test accuracy in the scratch model also shows more volatility, frequently oscillating between 70-85%. This comparison demonstrates the advantages of transfer learning - the pre-trained model not only converges faster but also achieves more stable and generally higher validation and test accuracies, suggesting it has learned more robust and generalizable features.

1.4 Overall classification accuracy on the testing set:

I have used the wandb.ai platform for reporting and plotting overall accuracy and class-wise accuracy instead of Tensorboard.
I have uploaded the class-wise and overall accuracy and loss metrics in the GitHub Link.

Training from Scratch.

Figure 3: The above figure shows the Training Loss/Accuracy, Validation Loss/Accuracy, and Test Loss/Accuracy. The training accuracy is improving and training loss is decreasing over 40 epochs. The validation accuracy is also increasing. However, the validation loss is unstable and fluctuating similar to test loss and test accuracy. Since this is from scratch, this can be because of random initialization which results in fluctuations in training.

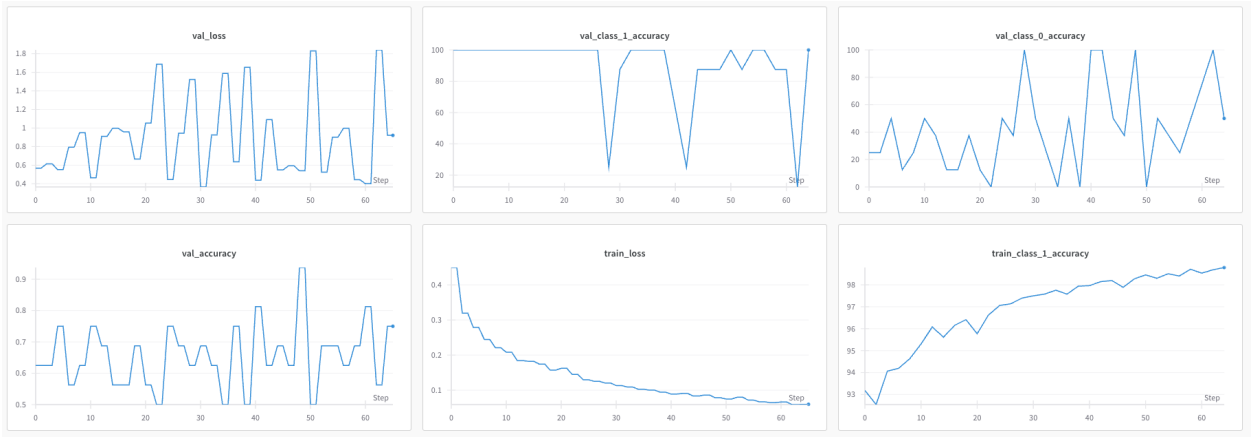

Figure 4: The above figure shows the Training Loss/Accuracy, Validation Loss/Accuracy, and Test Loss/Accuracy. This figure plots the class-wise validation and training loss. The training accuracy for class 1 is increasing and the training loss is decreasing as mentioned above. The validation accuracy of class 1 has subtle fluctuations whereas class 0 accuracy has more instability.

Training from Pre-trained ResNet18 weights.

Figure 5: The training loss curve shows a steady decrease, indicating that the model is learning effectively on the training data. The training accuracy for class 1 is consistently improving and nearing 100%. However, the validation loss exhibits significant fluctuations, suggesting that the model might be overfitting or struggling with generalization. Validation accuracy and class-specific validation accuracy also show high variance, implying inconsistency in performance across batches. This instability may be due to small batch sizes, data imbalance, or insufficient regularization. Further investigation, such as increasing the batch size, applying stronger regularization techniques, or using a learning rate scheduler, may help stabilize validation performance.

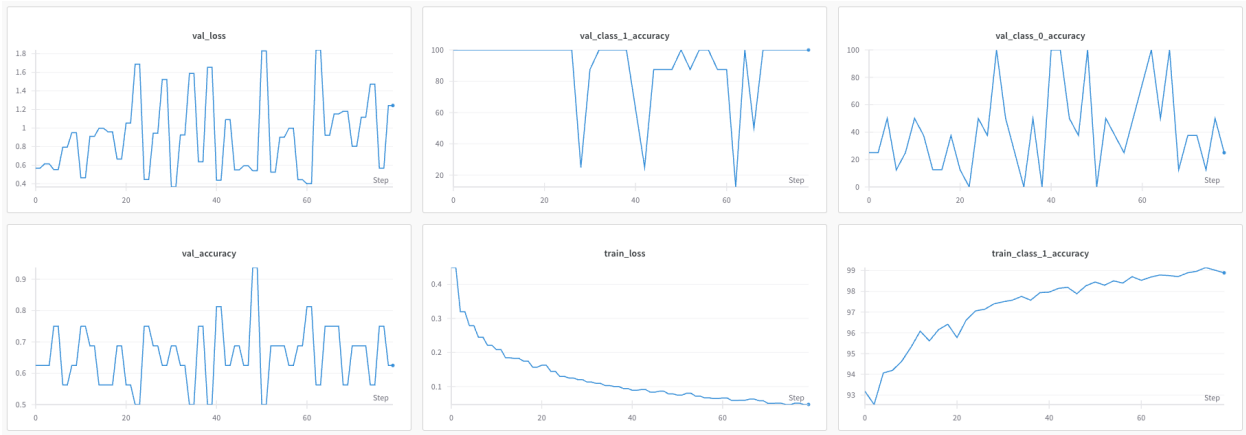

Figure 6: The class-wise validation and test loss show significant fluctuations, indicating that the model struggles with generalization, likely due to class imbalance or overfitting. While training loss steadily decreases and training accuracy reaches high values (0.95), the validation and test accuracy exhibit sharp variations, suggesting that the model performs well on certain classes but poorly on others. This instability implies potential issues with class difficulty, insufficient regularization, or optimization challenges. Regularization techniques, class re-balancing, and a learning rate scheduler could help stabilize performance and improve generalization across all classes.