Medical Image Computing CAP5516 Assignment#01

Ashmal Vayani, UCF ID: 5669011

February 11, 2025

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

1 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, and Validation loss curves on both Scratch (random initialization) and using pre-trained model weights on ResNet18.

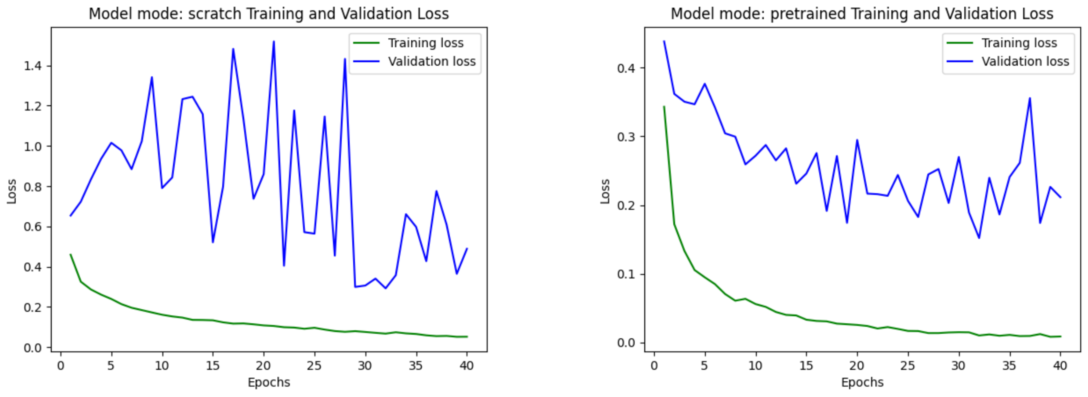
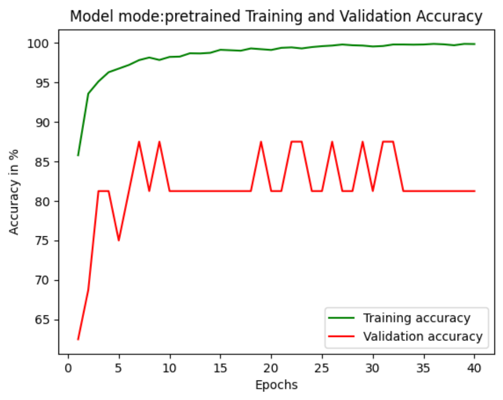


Figure 1: The above figure demonstrates that in the pre-trained model (Image 2), the validation loss starts at around 0.45 and gradually decreases to stabilize between 0.2-0.3, showing some minor fluctuations but maintaining a relatively steady pattern. The training loss smoothly decreases from 0.35 to nearly zero, indicating effective learning on the training data. In contrast, the model trained from scratch (Image 1) exhibits highly unstable validation loss behavior, with dramatic oscillations ranging from 0.4 to 1.4, particularly during the first 30 epochs. While its training loss curve similarly decreases to near zero, the extreme volatility in validation loss suggests the model struggles to find stable generalizable features. This comparison demonstrates the benefits of transfer learning through pretraining - it not only provides more stable validation performance but also achieves consistently lower loss values throughout training.

1.3 Training and Validation Accuracy curves:

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



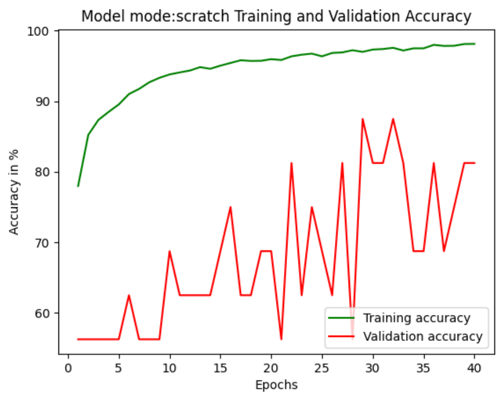


Figure 2: The pre-trained model (Image 1) demonstrates much more stable behavior, with validation loss starting around 0.45 and gradually decreasing to fluctuate between 0.2-0.3, while training loss smoothly decreases from 0.35 to nearly zero. In stark contrast, the model trained from scratch (Image 2) shows highly unstable validation loss behavior, with dramatic oscillations ranging from 0.4 to 1.4, particularly pronounced in the first 30 epochs. Both models achieve similarly low training loss (approaching zero).

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.

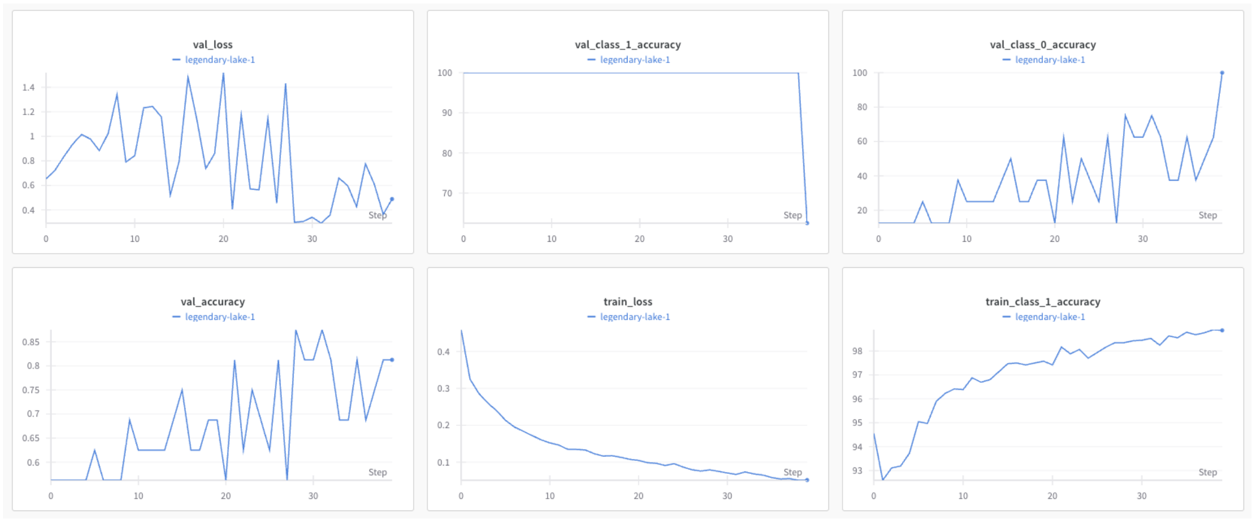


Figure 3: The above figure shows the Training Loss/Accuracy, Validation Loss/Accuracy, and Test Loss/Accuracy. The validation loss starts at around 0.4 and gradually decreases to stabilize between 0.2-0.25, though with noticeable fluctuations throughout training. The validation accuracy quickly improves in the early epochs and stabilizes around 0.8-0.85, indicating reasonably good model performance. Looking at class-specific performance, val class 1 accuracy maintains a steady 100% while val class 0 Accuracy shows more variation, fluctuating between 60-70%, indicating the class imbalance in the dataset. The training loss curve shows a steadily decreasing trend from 0.3 to nearly zero, while the training accuracy for class 1 (train class-1 Accuracy) gradually improves to reach above 99%, indicating strong learning on the training data.

Training from Pre-trained ResNet18 weights.

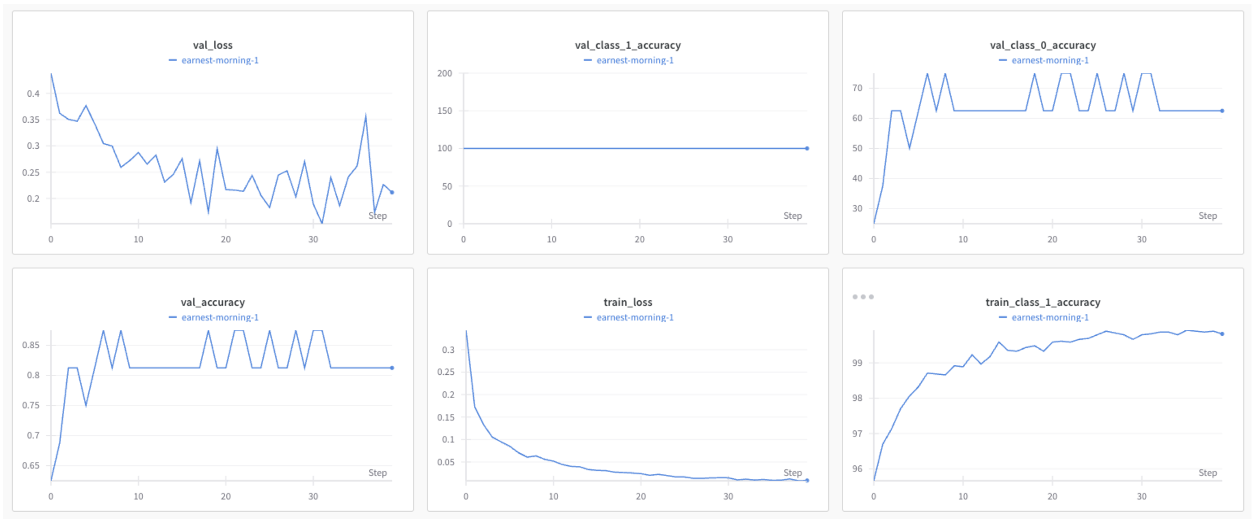


Figure 4: The graphs from the pre-trained experiment show more stable and improved performance compared to training from scratch. The validation loss demonstrates a steady decline from 0.4 to around 0.2-0.25, with some oscillations but generally more controlled behavior. The overall validation accuracy quickly rises and stabilizes at approximately 0.8-0.85, indicating good model generalization. Looking at class-specific performance, there's an interesting pattern where val class 1 Accuracy maintains a perfect 100% throughout training, while val class 0 Accuracy shows more variation, stabilizing around 60-70%. This disparity between class performances highlights the class imbalance in the dataset. The training loss exhibits expected behavior, smoothly decreasing from 0.3 to near zero, while train class 1 Accuracy shows excellent performance, climbing above 99%.

2 Data Augmentation

:

Transformation	**Hyperparameter(s)**
Grayscale Conversion	`num_output_channels=3`
Resize	`(224, 224)`
Random Horizontal Flip	`p=0.5`
Random Rotation	`degrees=10`
Random Affine	`degrees=0, translate=(0.1, 0.1)`
Color Jitter	`brightness=0.2, contrast=0.2, saturation=0.2`
Random Resized Crop	`224, scale=(0.8, 1.0)`
To Tensor	N/A
Normalize	`mean=(0.5,), std=(0.5,)`

Table 3: In my data augmentation pipeline, I first convert grayscale images to a three-channel format to match the model's input requirements. Then, I resize them to 224x224 pixels and apply random horizontal flipping with a 50% probability to introduce orientation variations. To further enhance diversity, I use a random rotation of up to ±10 degrees and affine transformations with slight translations (10% in both x and y directions), simulating different viewpoints. I also apply color jittering to adjust brightness, contrast, and saturation, making the model more robust to lighting variations. Additionally, I use random resized cropping, and scaling images between 80% and 100% of their original size to ensure diverse focal regions. Finally, I convert the images to tensors and normalize them with a mean of 0.5 and a standard deviation of 0.5 to stabilize training.

Training from Scratch.

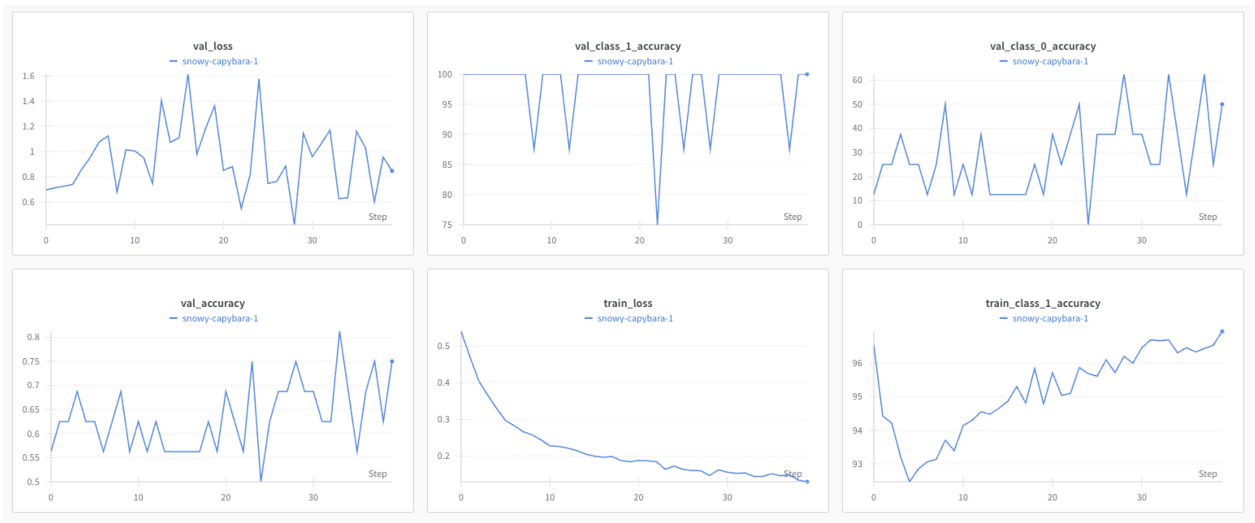


Figure 5: The graphs highlight the impact of extensive data augmentation on training from scratch, showing volatile dynamics. Validation loss fluctuates between 0.6 and 1.6, without clear convergence. Class-specific performance reveals that val class 1 accuracy stays high (95-100%) but occasionally drops to 75%, while val class 0 accuracy is highly unstable (0-60%). This suggests the model struggles with class imbalance or feature learning for class 0, despite augmentation techniques. Overall validation accuracy fluctuates between 55-80%, with more volatility in later epochs. Training loss decreases steadily from 0.5 to 0.15, and class 1 accuracy improves from 93% to 96%, but these gains don't translate to stable validation performance.

Training from Pre-trained ResNet18 weights.

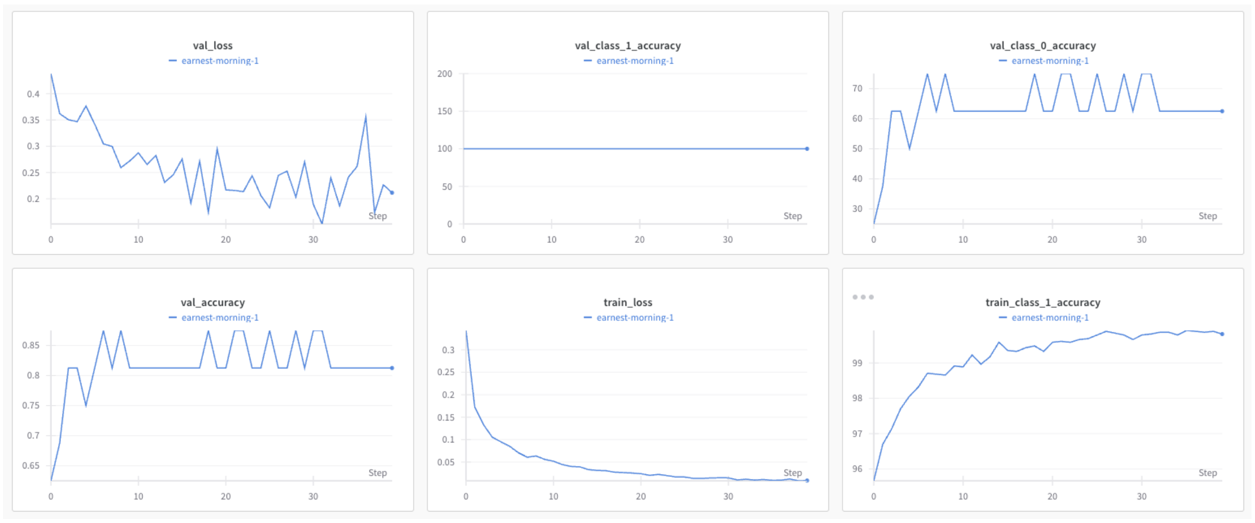


Figure 6: Despite pre-training and extensive data augmentation (grayscale conversion, random flips, rotations, translations, and color adjustments), the model shows significant instability. Validation loss fluctuates between 0.6 and 1.6, without clear convergence. While class 1 accuracy remains high (95-100%) with occasional drops, class 0 accuracy is unstable, ranging from 0-60%. Overall validation accuracy fluctuates between 55-80%, with volatility in later epochs. Training metrics show improvement, with training loss decreasing from 0.5 to 0.15 and class 1 accuracy improving from 93% to 96%. However, this progress doesn't translate to stable validation performance, indicating unresolved class imbalance and generalization challenges.

2.1 Data Augmentation - Training and Validation Accuracy curves:

The above figure shows the Training, and Validation Accuracy curve on both Scratch (random initialization) and using pre-trained model weights on ResNet18 with data augmentation.

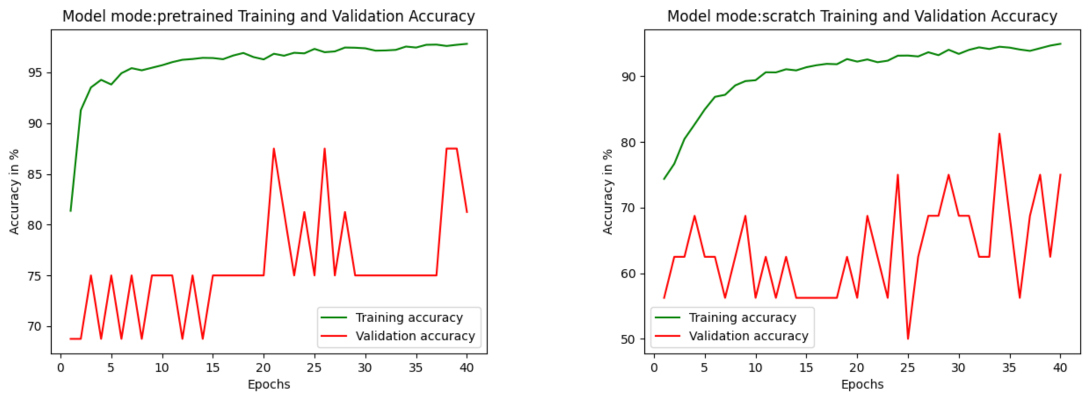


Figure 7: The pre-trained model (Image 1) demonstrates superior performance with training accuracy quickly reaching and stabilizing above 95%, while its validation accuracy maintains a higher baseline around 75% with occasional peaks reaching 87%. In contrast, the scratch-trained model (Image 2) shows lower overall performance, with training accuracy gradually improving to around 90% and validation accuracy fluctuating more dramatically between 50-80%. While both models exhibit a gap between training and validation accuracy indicating some overfitting, the pre-trained model shows a smaller gap and more stable validation performance.

2.2 Data Augmentation - Training and Validation Loss curves:

The above figure shows the Training, and Validation Loss curve on both Scratch (random initialization) and using pre-trained model weights on ResNet18 with data augmentation.

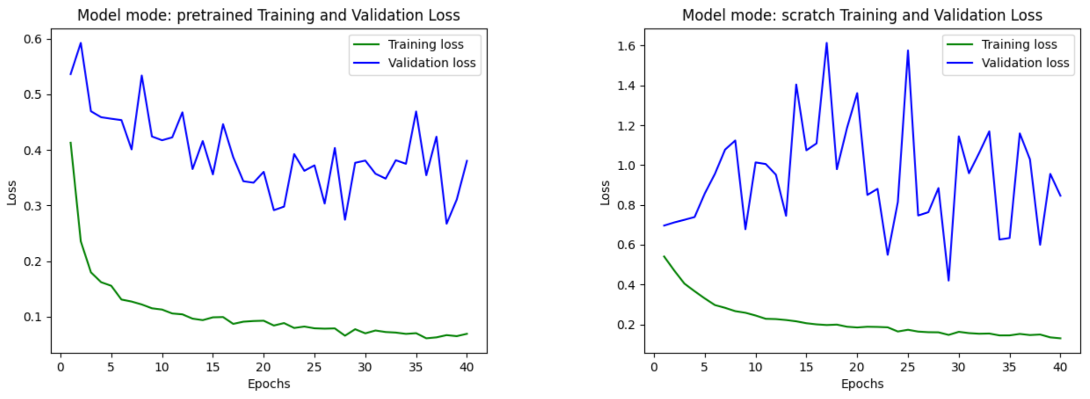


Figure 8: The pre-trained model (Image 1) demonstrates superior performance, with both training and validation loss values significantly lower throughout the training process. The validation loss closely tracks the training loss, indicating effective generalization and minimal overfitting. Additionally, the smoother loss curves suggest more stable training dynamics. In contrast, the scratch-trained model (Image 2) exhibits a typical pattern of overfitting, where training loss steadily decreases, but validation loss fluctuates considerably and remains much higher, highlighting poor generalization. The erratic nature of the validation loss curve further suggests instability in training. These results emphasize the advantage of using pre-trained weights, which not only accelerate convergence but also improve overall model robustness and generalization.

3 Qualitative Examples for wrongly predicted Samples

3.1 Random Initialization.

I show some qualitative examples of the failure cases of the training by randomly initializing the ResNet18 weights.

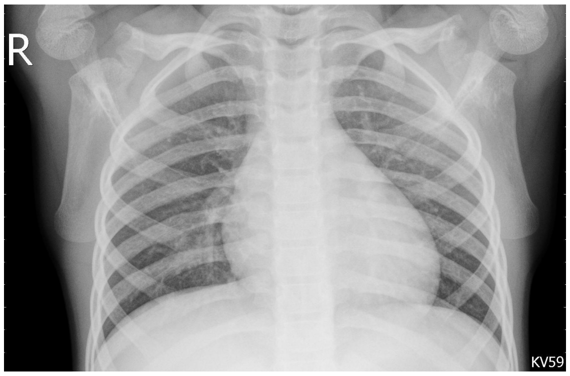


Figure 9: True Label: NORMAL, Predicted Label: PNEUMONIA

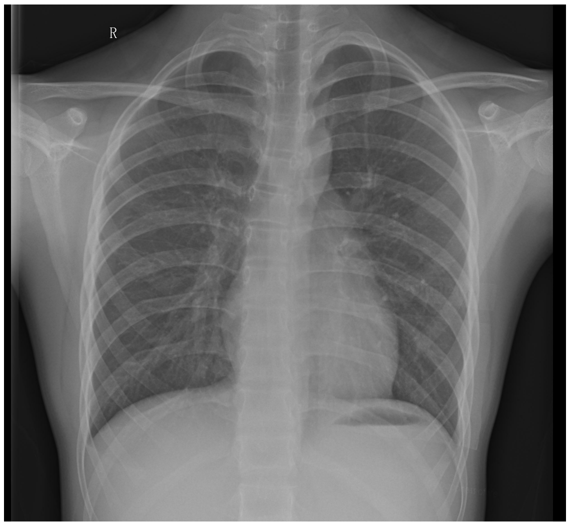


Figure 10: True Label: NORMAL, Predicted Label: PNEUMONIA

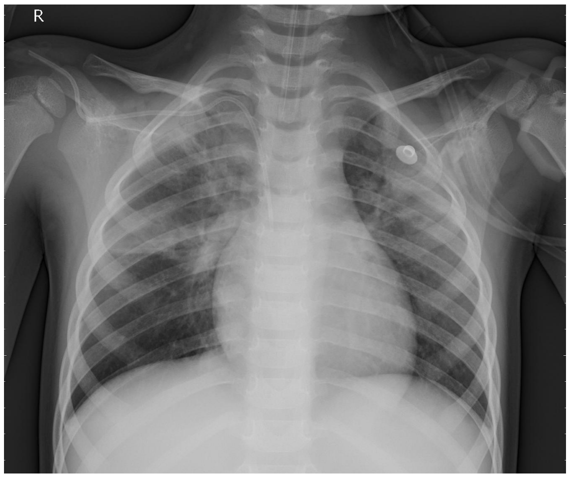


Figure 11: True Label: NORMAL, Predicted Label: PNEUMONIA

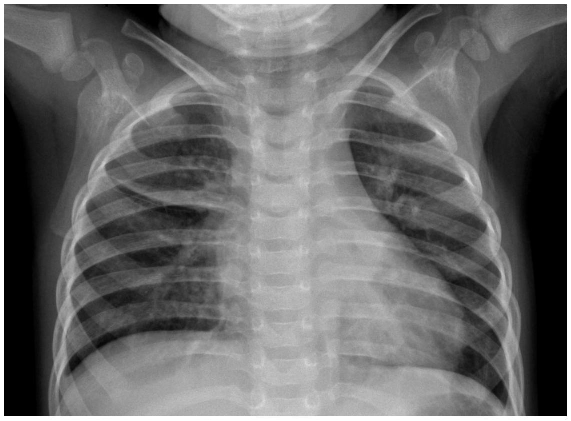


Figure 12: True Label: PNEUMONIA, Predicted Label: NORMAL

3.2 Pre-Trained weights of ResNet18.

I show some qualitative examples of the failure cases of the training on top of the pre-trained weights of the ResNet18 model.

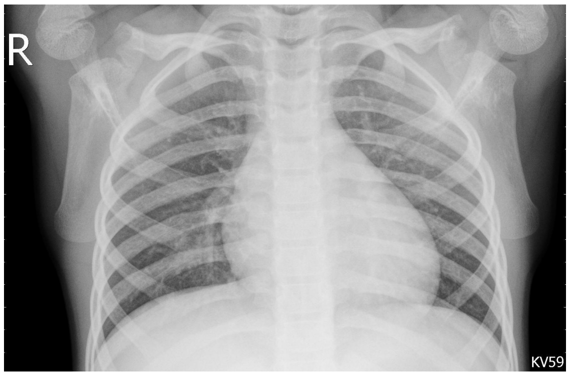


Figure 13: True Label: NORMAL, Predicted Label: PNEUMONIA

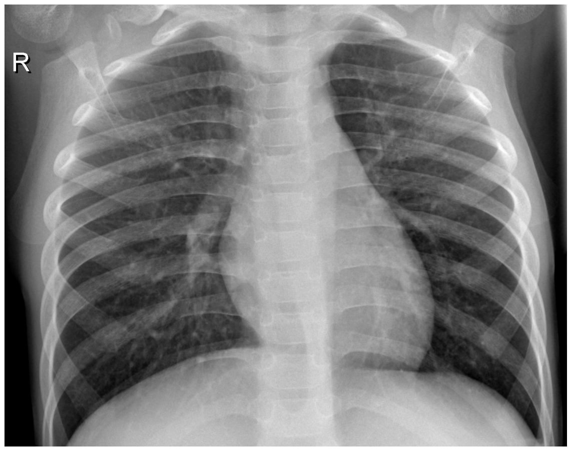


Figure 14: True Label: NORMAL, Predicted Label: PNEUMONIA

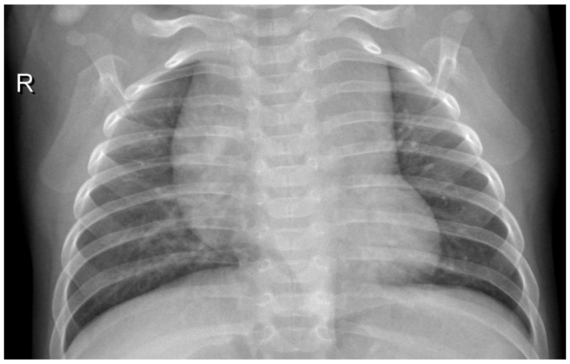


Figure 15: True Label: NORMAL, Predicted Label: PNEUMONIA

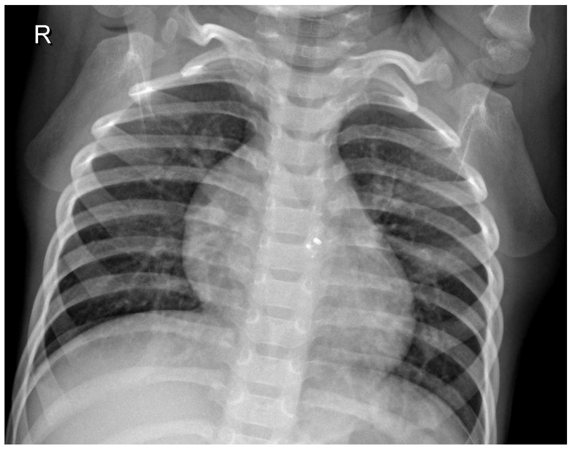


Figure 16: True Label: PNEUMONIA, Predicted Label: NORMAL