Deep Learning Lab Report

BADEREDDINE Haitham

April 25, 2025

1 Part 1: Transfer Learning

1.1 Introduction

This report covers the fourth lab of the Deep Learning course, focusing on transfer learning. We used ShuffleNet and EfficientNet models, and present their performance metrics. Transfer learning involves leveraging knowledge gained from training on one task or dataset to improve learning performance on a related but different task or dataset.

1.2 Dataset

The data consists of Chest X-rays from Kaggle, containing 21,165 samples. We used stratified hold out to maintain class balance.

1.3 Architectures Summaries

1.3.1 EfficientNet

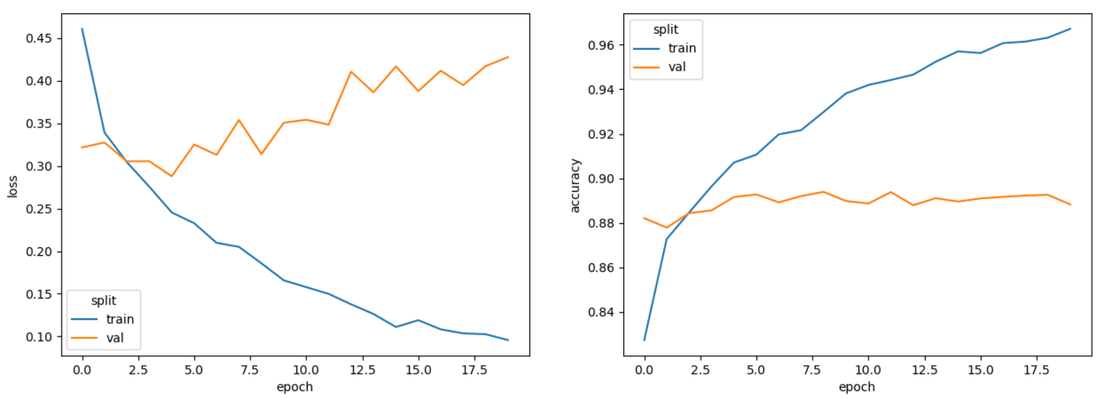


Figure 1: Accuracy And Loss for EfficientNet depth 0

depth = 0

```
================================================================
Total params: 5,846,144
Trainable params: 1,838,596
Non-trainable params: 4,007,548
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 173.68
Params size (MB): 22.30
Estimated Total Size (MB): 196.55
----------------------------------------------------------------
```

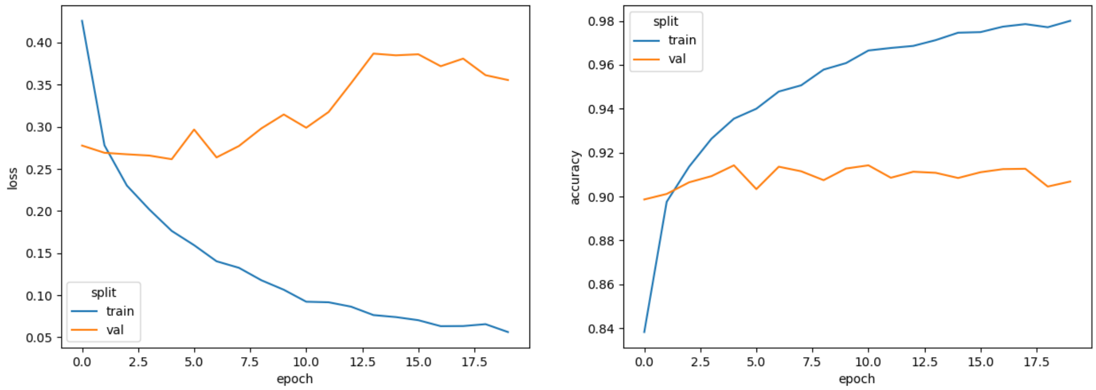


Figure 2: Accuracy And Loss for EfficientNet depth 1

depth = 1

```
================================================================
Total params: 5,846,144
Trainable params: 2,250,756
Non-trainable params: 3,595,388
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 173.68
Params size (MB): 22.30
Estimated Total Size (MB): 196.55
----------------------------------------------------------------
```

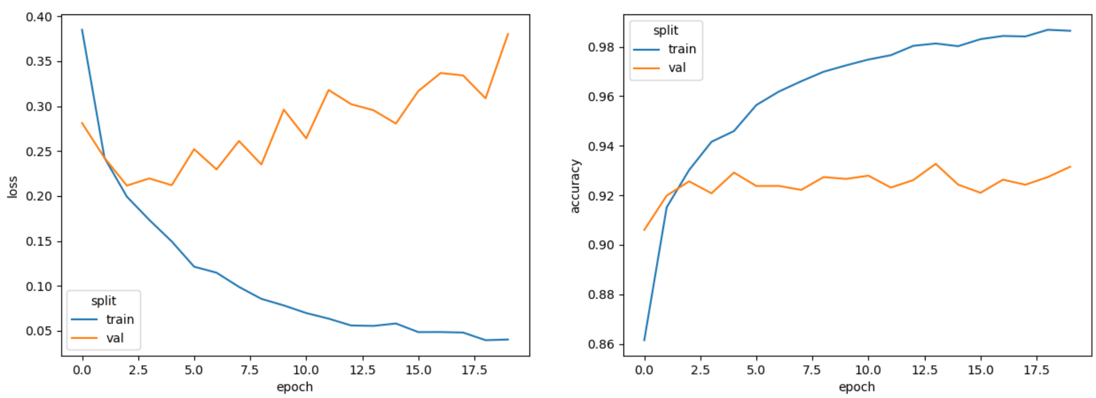


Figure 3: Accuracy And Loss for EfficientNet depth 2

depth = 2

```
================================================================
Total params: 5,846,144
Trainable params: 2,967,988
Non-trainable params: 2,878,156
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 173.68
Params size (MB): 22.30
```

```
Estimated Total Size (MB): 196.55
----------------------------------------------------------------
```

1.3.2 ShuffleNet

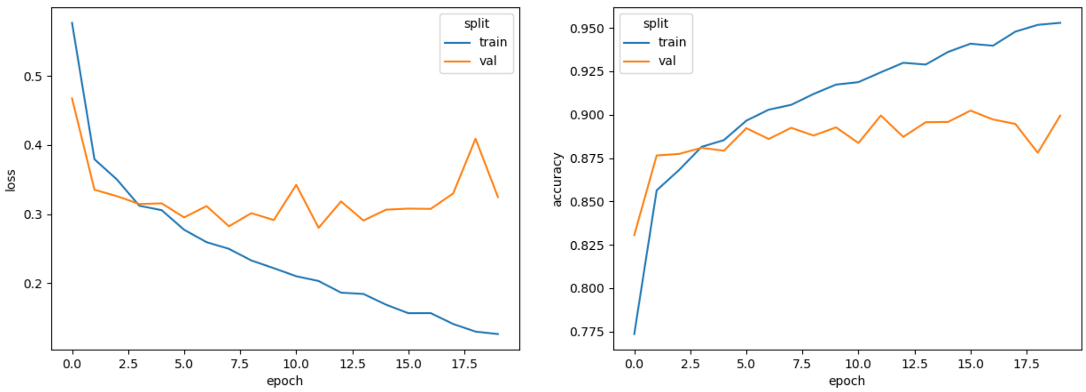


Figure 4: Accuracy And Loss for ShuffleNet depth 0

depth = 0

```
================================================================
Total params: 2,830,056
Trainable params: 1,576,452
Non-trainable params: 1,253,604
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 47.97
Params size (MB): 10.80
Estimated Total Size (MB): 59.34
----------------------------------------------------------------
```

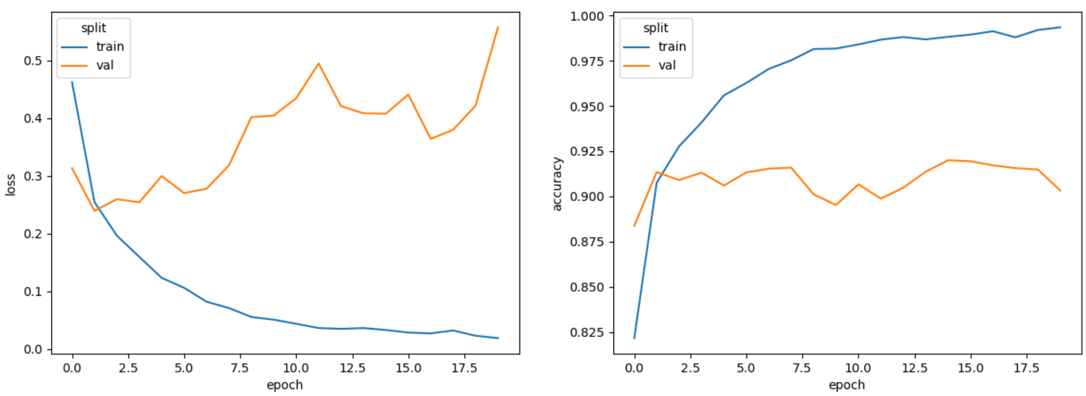


Figure 5: Accuracy And Loss for ShuffleNet depth 1

depth = 1

```
================================================================
Total params: 2,830,056
Trainable params: 2,053,636
```

```
Non-trainable params: 776,420
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 47.97
Params size (MB): 10.80
Estimated Total Size (MB): 59.34
----------------------------------------------------------------
```

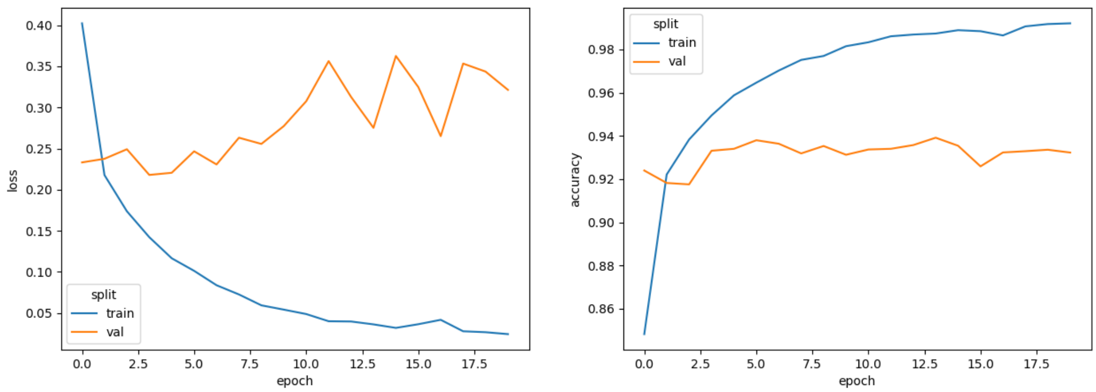


Figure 6: Accuracy And Loss for ShuffleNet depth 2

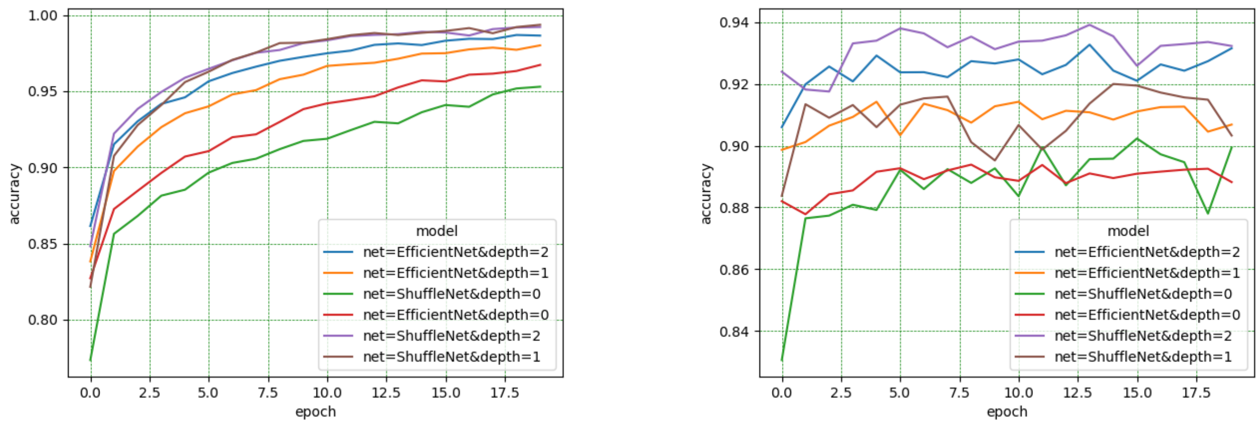
depth = 2

```
================================================================
Total params: 2,830,056
Trainable params: 2,554,988
Non-trainable params: 275,068
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 47.97
Params size (MB): 10.80
Estimated Total Size (MB): 59.34
----------------------------------------------------------------
```

1.4 Results

The following table shows the performance metrics of the different models:

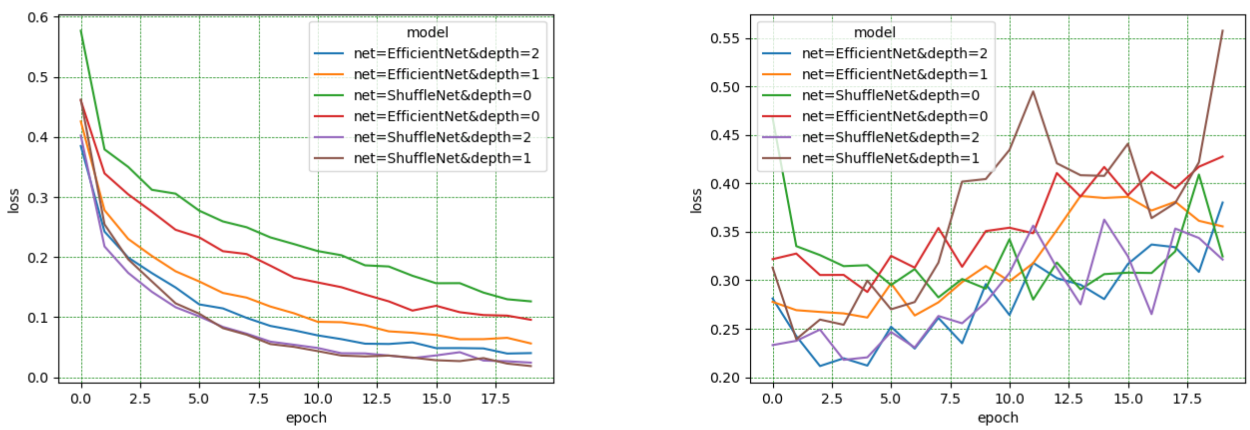
- depth 0: (the last three fully connected)
- depth 1: (one convolutional layers + the last three fully connected)
- depth 2: (two convolutional layers + the last three fully connected)



(a) Training Accuracy

(b) Validation Accuracy

Figure 7: Accuracy plots for different models



(a) Training Loss

(b) Validation Loss

Figure 8: Loss plots for different models

ShuffleNetV2x0.5			
Metric	**Depth 0**	**Depth 1**	**Depth 2**
Accuracy	0.8995	0.9036	0.9323
F1-Score	0.9063	0.9104	0.9390
Precision	0.9143	0.9134	0.9335
Recall	0.8988	0.9131	0.9448
AUC	0.9810	0.9829	0.9899

EfficientNet-B0			
Metric	**Depth 0**	**Depth 1**	**Depth 2**
Accuracy	0.8885	0.9068	0.9313
F1-Score	0.8994	0.9159	0.9376
Precision	0.9073	0.9095	0.9459
Recall	0.8920	0.9231	0.9304
AUC	0.9769	0.9848	0.9878

Table 1: Comparative Performance Metrics of ShuffleNetV2x0.5 and EfficientNet-B0 Models at Different Depths

1.5 Discussion

- **Accuracy and F1-Score**: Models with depth of 2 consistently achieve the highest accuracy and F1-Score values, surpassing 0.93 and 0.937, respectively, indicating strong overall performance.
- **Precision and Recall**: Models with depth of 2 demonstrate high precision and recall values, suggesting effective identification of positive instances with minimal false positives.

- **Area Under the Curve (AUC)**: All models exhibit AUC values exceeding 0.97, indicating strong discrimination ability, with Models with depth of 2 consistently achieving the highest values, surpassing 0.987.

ShuffleNet2 and EfficientNet2 emerge as the top performers across all metrics, showcasing their robustness and effectiveness for the given task.

1.6 Conclusion

Increasing the number of fine-tuned layers adds complexity to the model. This complexity may allow the model to capture more intricate patterns in the data, potentially leading to better performance, especially if the task at hand is complex and requires high-level representations. However, overly complex models may also be prone to overfitting, especially if the dataset is small or noisy, and increases the required computational resources.

2 Part 2: Image Segmentation Using U-Net

2.1 Introduction

Image segmentation is the process of partitioning an image into multiple segments or regions, typically to locate objects and boundaries. Unlike classification (which labels the entire image) or object detection (which draws boxes around objects), segmentation provides pixel-level understanding.

2.2 What is U-Net?

U-Net is a convolutional neural network designed for biomedical image segmentation. It is highly effective due to its encoder-decoder structure with skip connections.

2.3 Dataset

The data consists of Chest X-rays from Kaggle, containing 21,165 samples. We used stratified hold out to maintain class balance.

2.4 U-Net Architecture

U-Net is a symmetric encoder-decoder network with skip connections. It's composed of three main building blocks: DoubleConv, Down, and Up, culminating in a final output convolution.

2.4.1 DoubleConv Module

This block consists of two 3×3 convolutions, each followed by Batch Normalization and ReLU activation.

Purpose: Extracts high-level features while maintaining spatial information.

2.4.2 Down Block (Encoder Path)

This block applies a MaxPooling layer followed by a DoubleConv block.

Purpose: Reduces spatial dimensions and increases feature depth.

2.4.3 Up Block (Decoder Path)

This block upsamples the input and concatenates it with the corresponding encoder feature map before applying DoubleConv.

Purpose: Reconstructs spatial resolution and combines low-level encoder features for better localization.

2.4.4 Architecture Flow Summary

Stage	Channels	Operation
Input	1 channel	256×256 grayscale image
Encoder Block 1	64	Conv → BN → ReLU ×2
Encoder Block 2	128	MaxPool → DoubleConv
Encoder Block 3	256	MaxPool → DoubleConv
Encoder Block 4	512	MaxPool → DoubleConv
Bottleneck	1024	MaxPool → DoubleConv
Decoder Block 1	512	Upsample → Concatenate → DoubleConv
Decoder Block 2	256	Upsample → Concatenate → DoubleConv
Decoder Block 3	128	Upsample → Concatenate → DoubleConv
Decoder Block 4	64	Upsample → Concatenate → DoubleConv
Output	1 channel	1×1 Conv (Logits output)

Table 2: U-Net architecture summary

2.5 Evaluation Metrics

2.5.1 Dice coefficient (F1 score)

The Dice Score measures how similar two sets are — typically:

- the predicted segmentation mask
- and the ground truth mask

It ranges from 0 to 1:

- 1 means perfect overlap (prediction is identical to the ground truth)
- 0 means no overlap at all

2.5.2 Formula

$$\text{Dice Score} = \frac{2 \times |A \cap B|}{|A| + |B|} \tag{1}$$

Where:

- A is the set of predicted positive pixels
- B is the set of actual (ground truth) positive pixels
- $|A \cap B|$ is the number of correctly predicted positive pixels (true positives)

In terms of TP, FP, FN:

$$\text{Dice Score} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2}$$

Why Dice Score is Useful?

- Handles imbalanced data well (unlike plain accuracy)
- Used heavily in medical imaging, semantic segmentation, etc.
- It's sensitive to both false positives and false negatives

2.6 Results

2.6.1 Training & Validation Loss

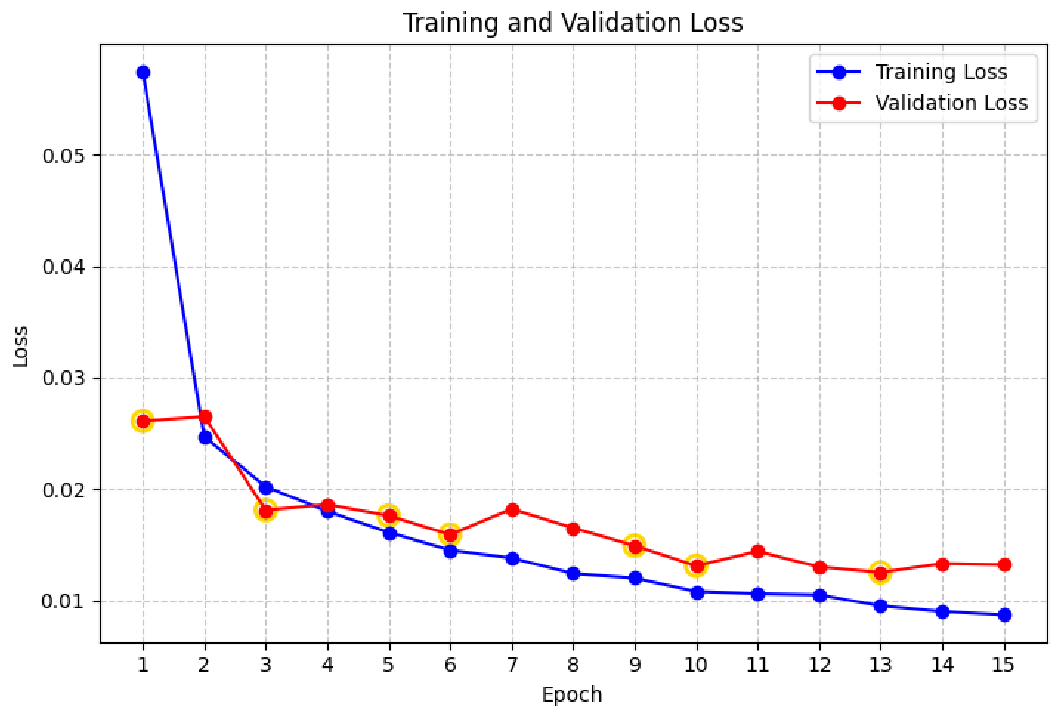


Figure 9: Training and validation loss curves

2.6.2 Dice Coefficient

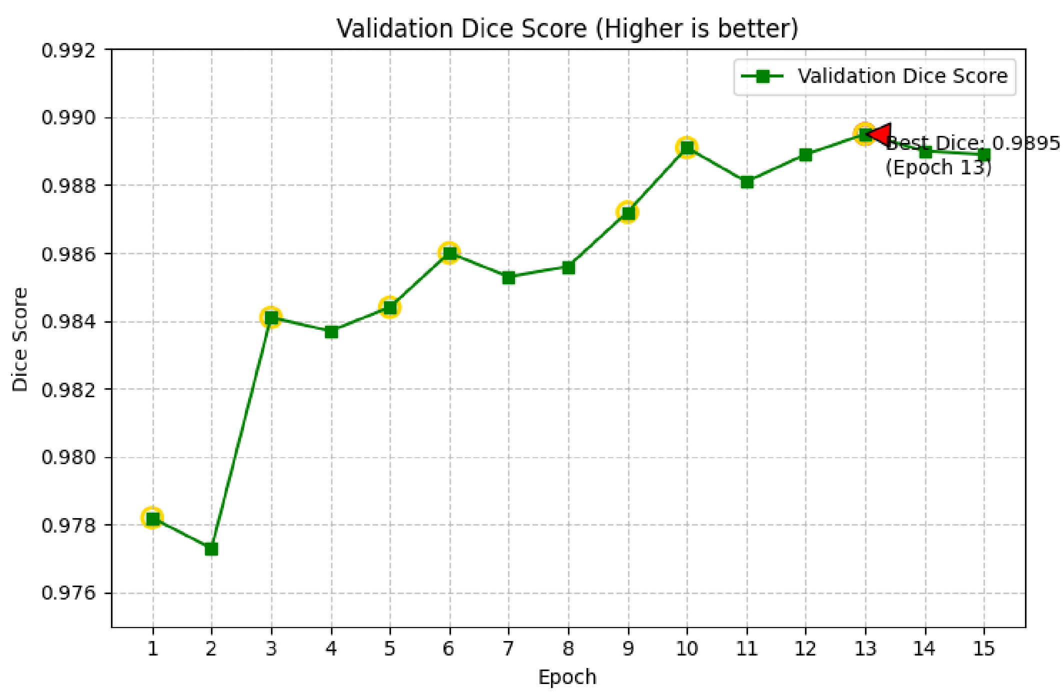


Figure 10: Dice coefficient over training epochs

2.6.3 Sample Predictions

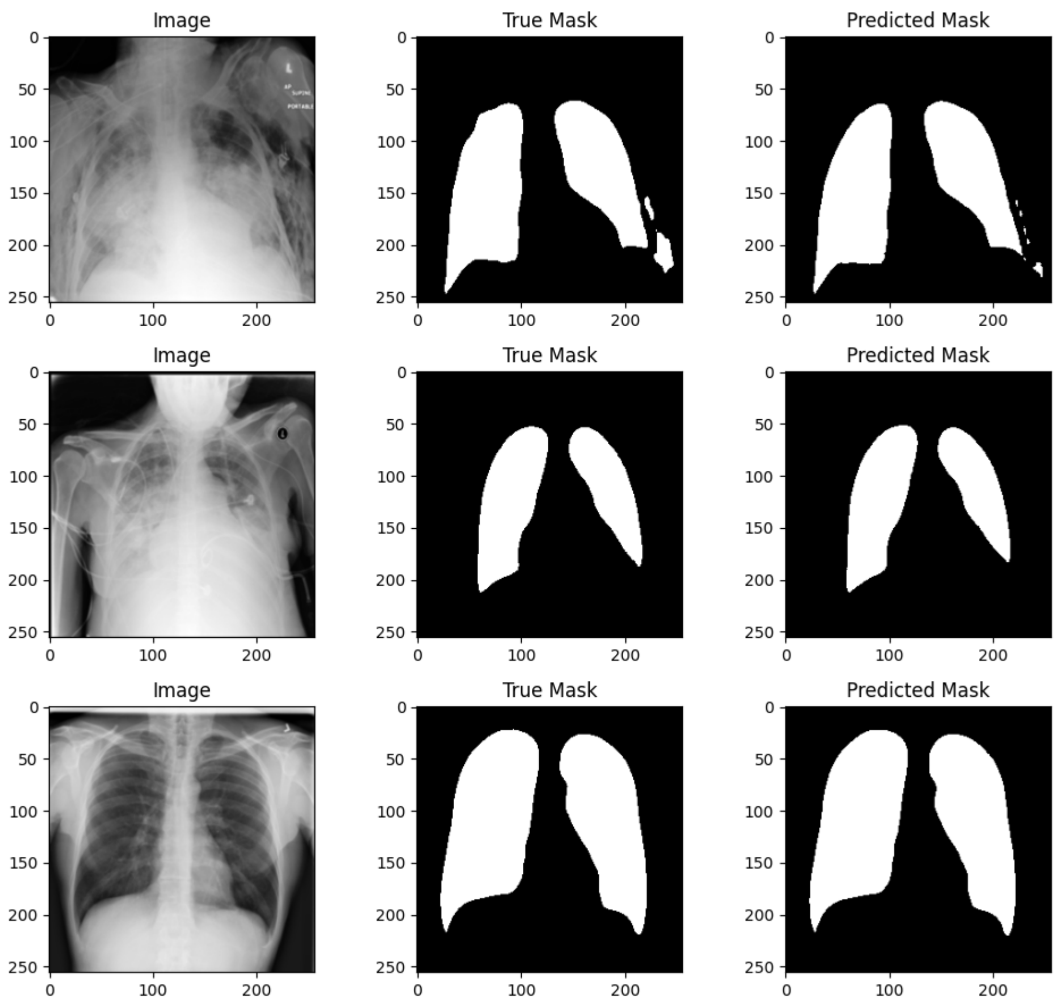


Figure 11: Sample segmentation predictions

2.7 Conclusion

U-Net remains one of the most popular and effective architectures for image segmentation tasks, particularly when dealing with limited training data. Its symmetric encoder-decoder structure with skip connections provides both context and localization capabilities, making it versatile across many domains. Modern variations continue to improve upon the original design while maintaining its core principles.