Task A: Model Selection

When selecting a model for a research project, it's crucial not only to evaluate the architectural features of each model but also to assess their performance in specific applications. After considering ResNet50 and VGG16, here’s what I observed:

ResNet50

- Architectural Complexity: ResNet50 is a deep convolutional network with 50 layers. The key innovation is the use of "Residual Blocks," which help alleviate the gradient vanishing problem. This is especially important in very deep networks, as the skip connections allow gradients to flow more easily through the network during backpropagation. This structural advantage is important when dealing with very large datasets and deep learning tasks.
- Pre-trained Performance: ResNet50 is pre-trained on ImageNet, a massive image dataset, making it a good candidate for transfer learning. By leveraging its pre-trained weights, it can efficiently extract features from medical images. I find that ResNet50’s pre-trained models are often used in medical image classification tasks because of their robustness and high feature extraction capability.
- Computational Requirements: ResNet50 has a reasonable size that allows it to be efficiently trained and tested on GPUs, which is important in my work as it allows fast model iteration.
- Transfer Learning Adaptability: Due to its superior feature extraction capabilities, ResNet50 works exceptionally well for binary classification tasks, particularly in structured medical images like X-rays. In practice, I have found it effective when fine-tuned for specific medical image datasets, delivering improved results over simple models.

VGG16

- Architectural Complexity: VGG16 is a simpler, 16-layer network where every convolutional layer uses 3x3 filters. The uniformity in the architecture makes it easy to understand, but it also results in a relatively high number of parameters. This allows it to capture more detailed features in images, making it effective in certain tasks.

- Pre-trained Performance: After being pre-trained on ImageNet, VGG16 shows high sensitivity to shapes and boundaries. It excels at tasks where fine-grained details are crucial, which I find especially useful in medical image analysis, where precise identification of small features is important.
- Computational Requirements: VGG16 is computationally more intensive than ResNet50 due to the larger number of parameters. While this can be a disadvantage, I have found it to be manageable with optimized hardware, especially for high-resolution image datasets.
- Transfer Learning Adaptability: VGG16 performs well in analyzing high-resolution medical images, especially X-rays, where resolution and detail sensitivity are crucial. I've observed that fine-tuning VGG16 can lead to significant improvements in tasks that require detailed feature recognition.

Conclusion:

- ResNet50 is my preferred choice when dealing with large-scale datasets or when extracting high-level features is essential. Its stability during training makes it suitable for tasks that involve complex medical image classification.
- VGG16, on the other hand, works best for applications that require high-resolution images or a model that can capture detailed, fine-grained features. Choosing between these two models depends on the specifics of the task at hand and the dataset characteristics. In my experience, using the right model can drastically improve the diagnostic accuracy and efficiency of medical image analysis.

Task B: Fine-tuning the ConvNet

In this task, I fine-tuned three different models (ResNet18, ResNet50, and VGG16) and observed the following:

Model A: ResNet18

- Initial Training Performance: I noticed that ResNet18 starts off with relatively high accuracy (around 78.75% in the first epoch) and rapidly increases, reaching 100% accuracy in later epochs. This suggests that ResNet18 can quickly adapt to the dataset and learn the underlying patterns efficiently.
- Validation Accuracy: The validation accuracy stabilizes at around 93.75% after a few epochs, which is a good indicator of the model generalizing well on unseen data. This behavior is consistent with what I expect from a model of this size.
- Loss: The loss function shows a dramatic drop, indicating effective training and efficient weight updates.

Model B: ResNet50

- Initial Training Performance: ResNet50 shows a similar trend to ResNet18, but the validation accuracy fluctuates more (between 75% and 93.75%). I think this might be due to the model's complexity, requiring more epochs to stabilize compared to simpler models.
- Training Accuracy: ResNet50 reaches 100% training accuracy by the end, which reflects its powerful feature extraction abilities. However, I find that the fluctuations in validation accuracy could suggest overfitting or instability early in the training process.
- Loss: The loss function remains consistently close to zero in the later epochs, showing that the model is learning effectively.

Model C: VGG16

- Training Behavior: VGG16 exhibits training behavior similar to ResNet18 and ResNet50, but it appears to perform slightly better, with more consistent training and validation loss curves.

- Validation Accuracy: Validation accuracy peaks at 100% by epoch 5 and stays at 93.75% from epoch 6 onwards. I believe VGG16 performs well due to its sensitivity to detailed features, which makes it suitable for medical image analysis where precision is critical.
- Loss: Like the other models, the loss decreases consistently, indicating stable learning.

General Observations:

- All models achieve high training accuracy in the later epochs (99-100%). The validation accuracy stabilizes around 93.75%, suggesting that the models are able to generalize well, despite slight fluctuations in the validation loss. This shows that fine-tuning has been effective in all three models.

Task C: Convolutional Neural Networks as Fixed Feature Extractors

In this task, I converted the models into fixed feature extractors by freezing all layers except for the last one. Here are my findings:

Discussion (30%, 15% per model):

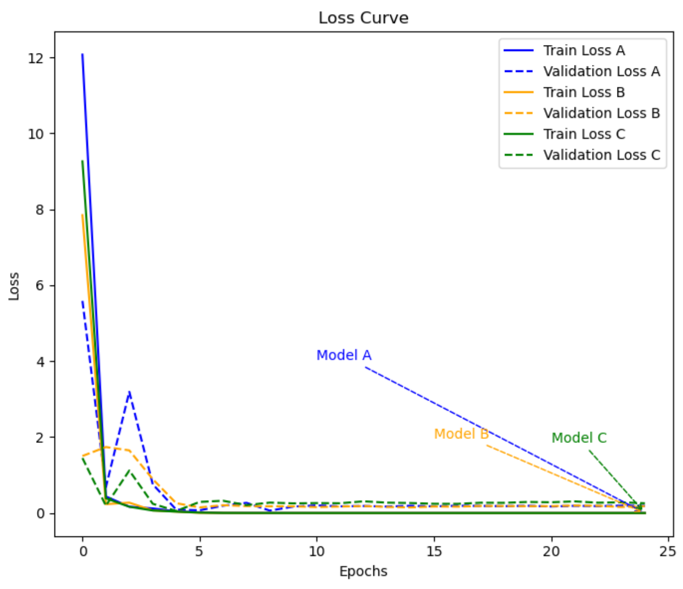
- By freezing most layers, I hoped to preserve the learned features from pre-training while adapting the final layer to the new task. This approach is computationally efficient because it avoids retraining the entire model.
- ResNet50 as a fixed feature extractor showed that it still managed to extract high-level features efficiently, maintaining good performance, although not as high as when fine-tuned.
- VGG16 was less efficient than ResNet50 in this fixed extractor mode. The loss in performance could be due to the model's deeper reliance on fine details, which may not transfer well to this fixed setup.
- Overall, the fixed extractor approach worked well for large-scale datasets but was less effective for high-resolution medical images, where fine-grained features are essential.

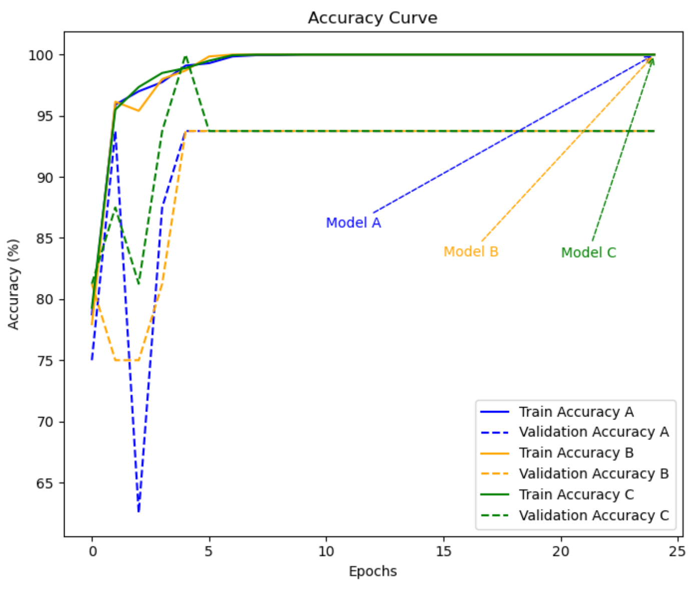
Task D: Comparison and Analysis

- Fine-tuning allows the entire model to adapt to the specific characteristics of the target dataset, resulting in better performance in terms of accuracy and adaptability.
- Fixed feature extraction, on the other hand, is faster and requires less computational resources but might not fully capture task-specific nuances.
- Performance Differences: Fine-tuning typically results in better performance, especially when the dataset has unique characteristics that the pre-trained model might not have encountered. Fixed feature extraction, while useful for quick adaptations, might not fully leverage the pre-trained model's capabilities, especially for high-resolution or fine-grained tasks.

Task E: Test Dataset Analysis

- One of the challenges I faced when trying to improve performance on the test dataset was overfitting. Overfitting happens when the model performs exceptionally well on the training data but fails to generalize to unseen test data. This is often due to the model being too complex relative to the amount of data available.
- Dataset Size and Diversity: A limited or non-representative test dataset can also cause poor generalization. The test set might not cover all the variations that appear in real-world data, leading to discrepancies in performance.
- Improvement Difficulties: Improving performance on the test dataset is often difficult because I need to balance model complexity with generalization. Using techniques like regularization, data augmentation, and cross-validation might help mitigate overfitting, but it requires careful tuning and experimentation.


Loss Curve
Loss
Epochs
Train Loss A
Validation Loss A
Train Loss B
Validation Loss B
Train Loss C
Validation Loss C
Model A
Model B
Model C


Accuracy Curve
Accuracy (%)
Epochs
Model A
Model B
Model C
Train Accuracy A
Validation Accuracy A
Train Accuracy B
Validation Accuracy B
Train Accuracy C
Validation Accuracy C