## Model Approach

The model approach I chose for this prediction task is a **Convolutional Neural Network (CNN)** architecture, which is well-suited for object classification tasks like this one, especially when dealing with image data. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images, making them highly effective for tasks such as detecting and classifying visual patterns.

## Complexity of the modeling approach

### 1. Convolutional Layers for Image Processing

#### ResNet-18 Architecture

The model is based on ResNet-18, a deeper and more advanced architecture compared to a traditional 3-layer CNN. While a 3-layer CNN consists of only three convolutional layers followed by batch normalization and max pooling, ResNet-18 has 18 layers, organized into residual blocks. These blocks use shortcut connections, allowing information to bypass certain layers and improving the flow of gradients during training. This mitigates the vanishing gradient problem, which can occur in deeper networks, and allows ResNet-18 to capture more complex and hierarchical features.

Unlike the 3-layer CNN, which may struggle to learn deeper representations of patterns such as textures and shapes, ResNet-18 is specifically designed to handle these challenges. It can learn more detailed features from the data without a significant increase in computational cost. ResNet-18 is, therefore, a better fit for complex tasks like skin lesion detection, where fine details and spatial hierarchies play a critical role in achieving high classification accuracy.

### Batch Normalization

Batch Normalization helps stabilize training by normalizing activations, which can make training deeper networks more efficient.

### Max Pooling

The use of pooling layers reduces the dimensionality of the feature maps and adds spatial invariance. This means that the model will focus on the most critical features from different areas of the image, reducing the computational burden.

## 2. Metadata Integration

### Fully Connected Layer for Metadata

The model incorporates a separate pathway to process metadata, which adds complexity because it requires the model to handle different data modalities. The metadata is processed through a simple fully connected network that reduces the dimensionality and learns meaningful representations from the metadata features.

### Concatenation

After processing both the image and metadata inputs, the model concatenates the extracted features. This step combines the learned representations from both modalities, adding complexity in terms of learning how to weight and utilize both feature sets in the final prediction.

## 3. Fully Connected Layers for Classification

After concatenating the image and metadata features, the model passes them through a fully connected layer to make a binary classification (output = 1 node). This final fully connected layer serves as the decision-making component of the model. While it's straightforward in terms of operations, the complexity comes from how well it can learn to combine the different feature sets (image and metadata).

## 4. Regularization

### Dropout Layer

To prevent overfitting, I've introduced dropout (50%). This adds complexity by randomly deactivating neurons during training, which forces the network to be more robust and prevents it from over-relying on specific neurons.

## 5. Binary Classification Task

The model is set up for binary classification with a sigmoid activation at the output. This reduces the final output to a single probability score between 0 and 1, suitable for binary labels (e.g., benign or malignant).

## Hyperparameters

### 1. Optimizer (Adam vs. SGD)

- The optimizer plays a critical role in determining how the model updates its weights. I evaluated both **Adam** and **SGD** to compare their efficiency in converging to an optimal solution.
    - **Adam**: Known for its adaptive learning rate, Adam often converges faster and requires less manual tuning. It is especially helpful when working with image data where feature gradients can vary.
    - **SGD**: While often slower to converge, **Stochastic Gradient Descent (SGD)** with momentum has been shown to provide better generalization and stability, especially for image classification tasks.
- I compared both optimizers to see which led to faster convergence and better generalization, particularly focusing on validation accuracy and loss. This comparison helped determine which optimizer was more suited to the task of skin lesion classification.

### 2. Learning Rate (0.001 vs. 0.0001)

- The learning rate controls how quickly the model updates its weights during training. If set too high, the model risks overshooting the optimal values, while a low learning rate

can result in slower convergence. To find the right balance, I evaluated learning rates of 0.001 and 0.0001:
    - 0.001: This is a commonly used starting point and typically provides a good balance between learning speed and stability. I tested this to see if it would allow the model to learn at a reasonable pace without being too aggressive.
    - 0.0001: A lower learning rate was explored to improve stability, especially when overfitting or erratic training behavior became evident with a higher rate. However, I found that this slower rate required significantly more epochs, making it less practical in my environment.
- Ultimately, I evaluated both rates to balance speed and stability, closely monitoring validation loss and accuracy over epochs. While 0.0001 improved stability, I found it less favorable due to the environmental constraints, which demanded longer training times.

## 3. Weight Decay

- Weight decay acts as an L2 regularization technique, penalizing large weights and helping prevent overfitting. In a complex model that handles both image and metadata, overfitting can be a concern, especially with limited medical imaging data. I included weight decay to ensure that the model generalizes well to unseen data.
- Weight decay was introduced to limit overfitting, particularly as the model began to improve on the training data. The goal was to maintain a balance between learning useful patterns and avoiding excessive reliance on specific features.

## 4. Batch Size (64 vs. 32)

- The batch size affects both the stability of gradient estimates and the memory consumption during training. A larger batch size (e.g., **64**) can speed up training but may lead to overfitting due to smoother gradient updates. In contrast, a smaller batch size (e.g., **32**) introduces more variability in gradient estimates, which can help the model generalize better by escaping local minima.
- I evaluated the impact of changing the batch size from **64 to 32** to observe how it affected both the training dynamics and the generalization of the model. The smaller batch size introduced more noise into the training process, potentially helping avoid overfitting and allowing the model to converge more effectively.
- By comparing the current model with previous versions, I found that using a batch size of 32 leads to more stable training and validation loss. However, this stability comes at the cost of slower training, requiring more iterations to achieve the same level of performance. While larger batch sizes tend to speed up the training process, the trade-off with batch size 32 is that it offers a smoother learning curve and helps avoid fluctuations

in loss, making it more reliable for tracking progress, especially in complex tasks like skin lesion classification.

## Model Performance Metrics

### 1. Validation Loss

Validation loss represents how well the model performs on unseen data, i.e., the model's generalization ability. It is the loss function (Binary Cross-Entropy in this case) calculated on the validation set after each epoch of training. Minimizing validation loss is crucial because the model should generalize well to new, unseen data—especially important for medical applications like skin lesion classification, where the consequences of poor generalization could lead to misdiagnosis. Validation loss helps detect overfitting. If validation loss increases while training loss decreases, the model is likely overfitting to the training data.A lower validation loss indicates that the model is learning well and is able to generalize to new images, ensuring that it can accurately classify skin lesions in real-world scenarios.

### 2. pAUC-aboveTPR

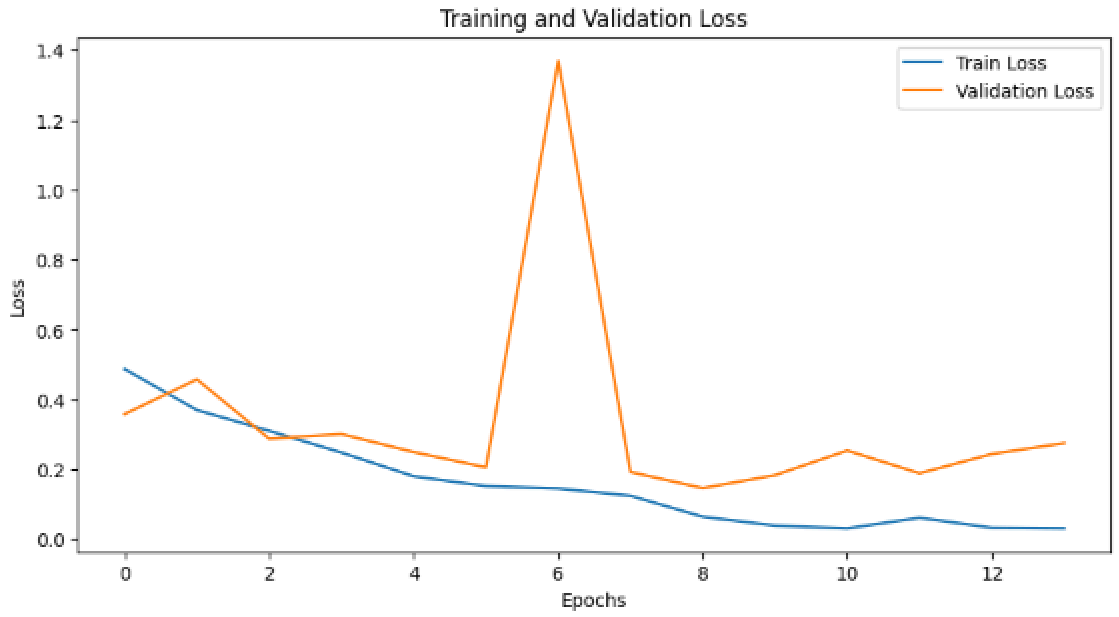
In skin lesion classification, using **partial AUC-aboveTPR** is essential because it allows the model to focus on maximizing sensitivity (True Positive Rate, TPR) while controlling the False Positive Rate (FPR). In medical diagnostics, especially in detecting skin cancer, minimizing false negatives (i.e., missed malignant lesions) is critical, as missing a malignant case could have severe consequences for the patient. Partial AUC-aboveTPR enables the evaluation of the model's performance within a specific range of TPR, such as ensuring at least 80% sensitivity—often a clinical standard. By doing so, it ensures that the model can reliably detect malignant cases while also maintaining an acceptable FPR, reducing unnecessary follow-ups or anxiety over benign lesions. This balance between sensitivity and precision aligns with real-world clinical needs, where early and accurate detection of malignant cases is paramount, but false positives must be kept under control to avoid overburdening healthcare systems and patients.

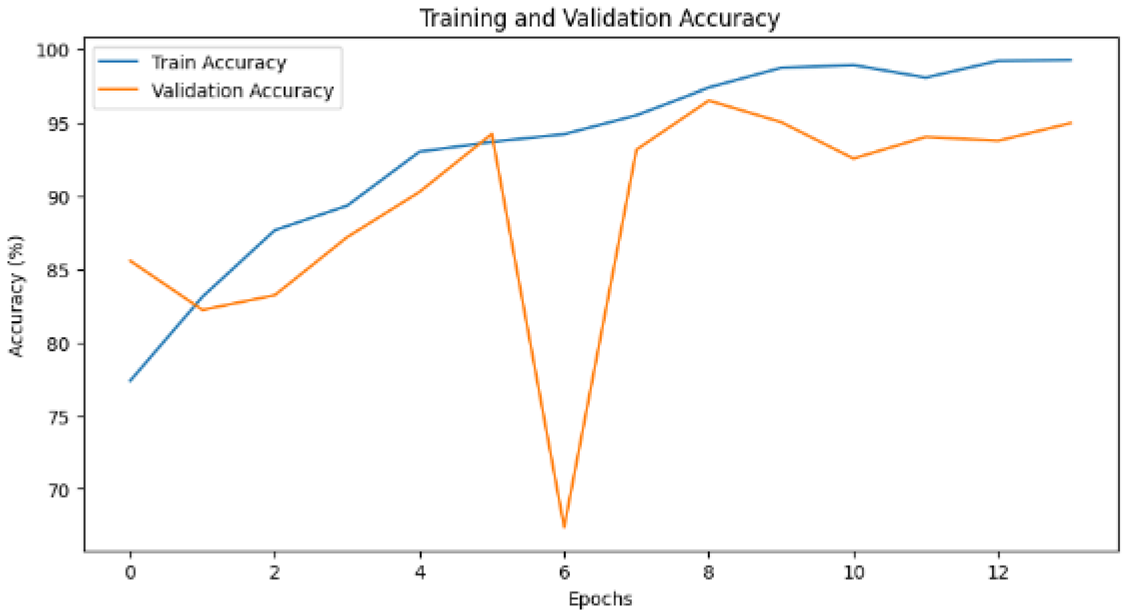
### 3. Accuracy

Accuracy is the percentage of correct predictions out of the total predictions made by the model. It is a simple ratio of correctly predicted instances (both benign and malignant) to the total number of instances. Accuracy provides an overall measure of how often the model is getting predictions right, which is still a key measure of success in classification tasks.While accuracy is straightforward, it can be misleading in the case of class imbalance (e.g., if there are far more benign cases than malignant). Therefore, while accuracy is tracked, it is evaluated in conjunction with ROC AUC to ensure that the model isn’t just performing well by predicting the majority class.

# Model Variations

## Model 1



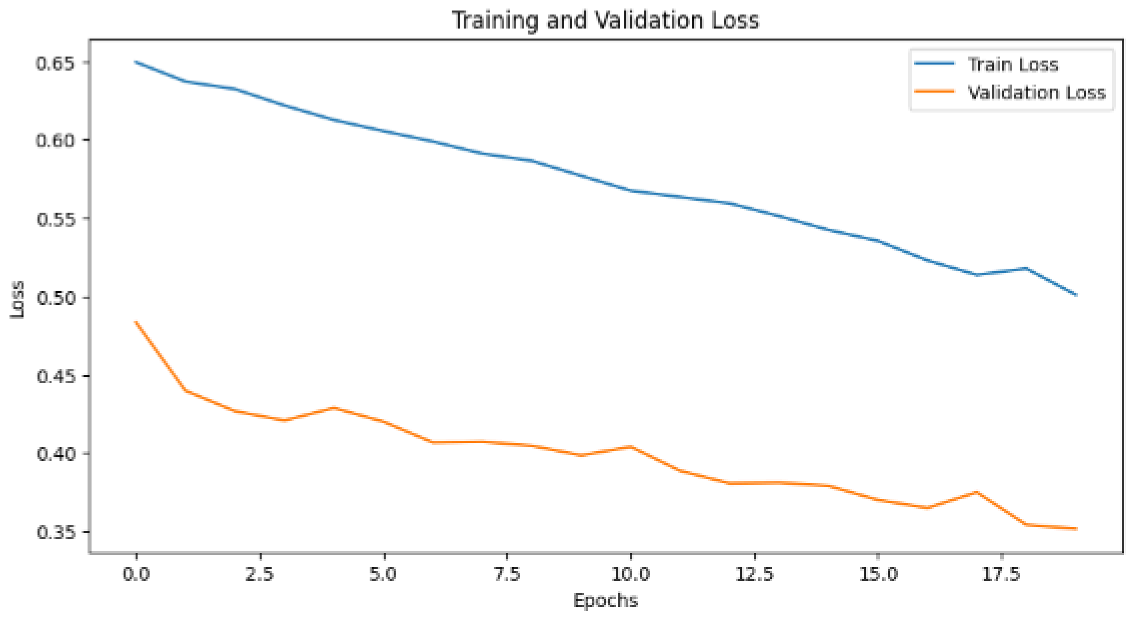


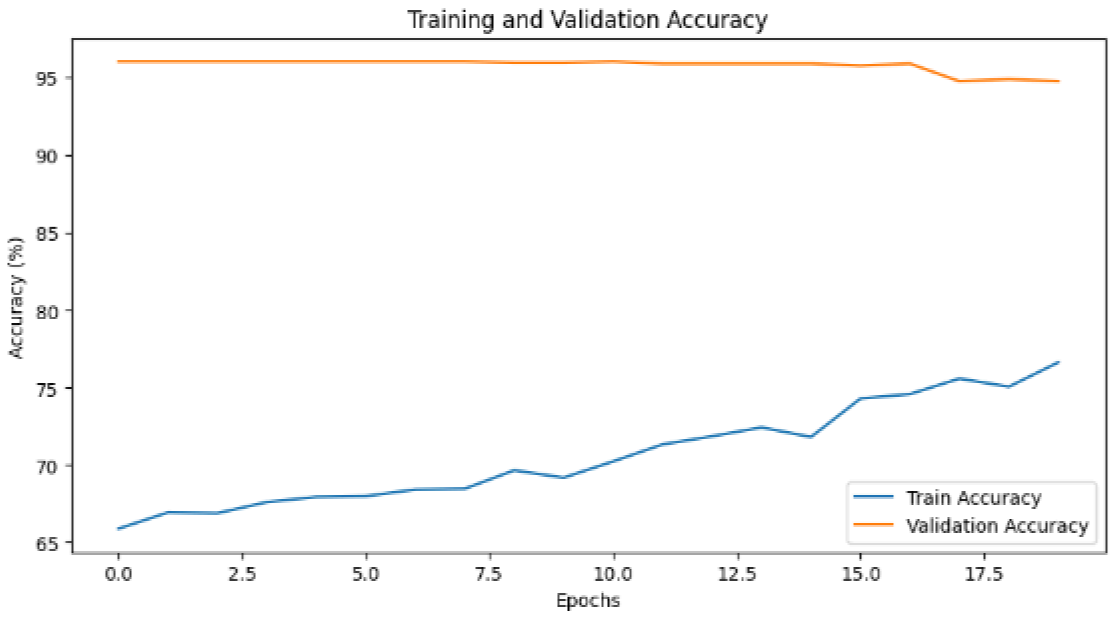
```
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.97      0.98      0.97      1431
     Class 1       0.35      0.31      0.32        59

    accuracy                           0.95      1490
   macro avg       0.66      0.64      0.65      1490
weighted avg       0.95      0.95      0.95      1490
```

Hyperparameter

- Optimizer Adam with learning rate of 0.001
- Binary Cross Entropy Loss function
- 20 epochs
- Batch size of 64

## Model 2





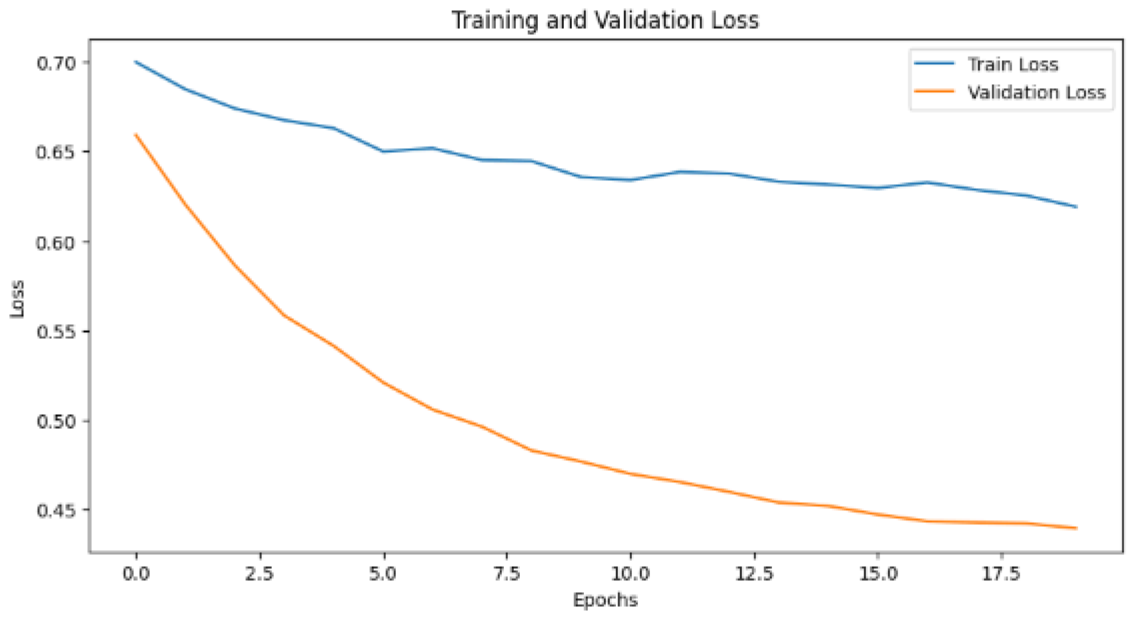
```
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.97      0.98      0.97      1431
     Class 1       0.31      0.27      0.29        59

    accuracy                           0.95      1490
   macro avg       0.64      0.62      0.63      1490
weighted avg       0.94      0.95      0.95      1490
```

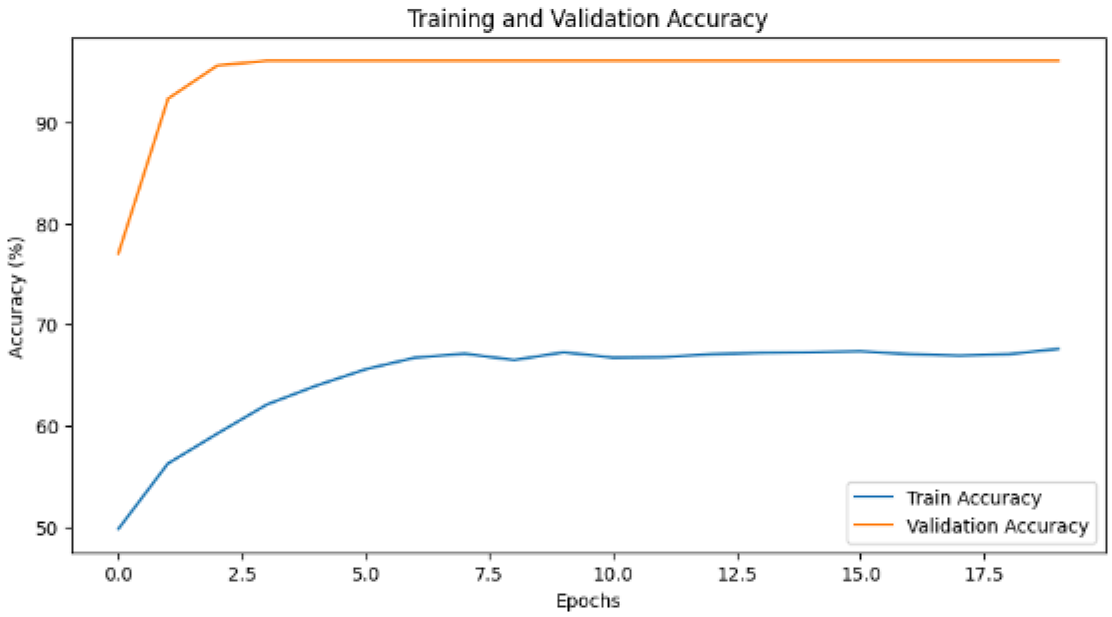
Hyperparameter

- Optimizer Stochastic gradient descent  with learning rate of 0.001
- Binary Cross Entropy Loss function

- 20 epochs
- Batch size of 64

## Model 3





```
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.96      1.00      0.98      1431
     Class 1       0.00      0.00      0.00        59

    accuracy                           0.96      1490
   macro avg       0.48      0.50      0.49      1490
weighted avg       0.92      0.96      0.94      1490
```

Hyperparameter

- Optimizer Stochastic gradient descent  with learning rate of 0.0001
- Binary Cross Entropy Loss function
- 20 epochs
- Batch size of 32

| Variation | Accuracy | Loss | pAUC-aboveTPR |
|---|---|---|---|
| Model 1 | 0.95 | Train Loss: 0.0645<br><br>Val Loss: 0.1464 | 0.1200 |
| Model 2 | 0.93 | Train Loss: 0.5011<br><br>Val Loss: 0.3516 | 0.0767 |
| Model 3 | 0.96 | Train Loss: 0.6192<br>Val Loss: 0.4397 | 0.0247 |

## Model Variation Analysis

I observed that Model 2's training and validation losses demonstrate a more consistent decline compared to Model 1, likely due to the switch from Adam to SGD, which may have provided more stable and effective learning. In contrast, Model 3 shows a slower reduction in its loss curve, indicating a more gradual learning process. This could be attributed to adjustments in hyperparameters or learning rate.

Given the dataset's imbalance, focusing solely on accuracy can be misleading. While precision for the majority class is high, it is significantly lower for the minority class. Therefore, metrics that better reflect the model's performance on the minority class should be prioritized. In this case, the competition organizer’s chosen performance metric, pAUC-aboveTPR, is more appropriate. Based on this metric, Model 1 outperforms both Model 2 and Model 3, delivering better results for this task.

However, it's important to consider the gap between training and validation losses. Without the restrictions of the current environment and computational limitations, I believe that with more iterations, Model 2 could outperform Model 1 in terms of overall performance.

## Conclusion

In conclusion, I will be selecting Model 1 as my final model for this week. The training and validation loss graphs, along with the accuracy graph, show promising improvement as the epochs progress. More importantly, Model 1 outperforms the other models in terms of the key performance metric, pAUC-aboveTPR, which is essential for this task. This metric's focus on sensitivity and false-positive control makes Model 1 the optimal choice. From the classification report, it's evident that Model 1 achieves a precision of 35% for Class 1, further highlighting its effectiveness. Additionally, this week's model architecture has proven to be significantly superior to last week's, with notable improvements in overall performance and effectiveness.