Model Approach

The model approach I chose for this prediction task is based on **EfficientNet**, a modern Convolutional Neural Network (CNN) architecture that has proven effective in handling complex image classification tasks, like the one at hand. EfficientNet is particularly suited for image data as it adaptively learns spatial hierarchies of features from input images, making it a strong choice for tasks that require detecting and classifying subtle visual patterns.

Complexity of the Modeling Approach

EfficientNet Architecture

The model is built upon EfficientNet, which is a more advanced architecture compared to the ResNet-18 model I used previously. EfficientNet is designed with a compound scaling method, balancing the network's depth, width, and resolution to achieve optimal performance while maintaining computational efficiency. This approach enables EfficientNet to learn more detailed and hierarchical features from images without a significant increase in computational cost, making it a better fit for tasks like skin lesion detection, where fine details play a critical role in achieving high classification accuracy.

In contrast to ResNet-18, which relies on residual blocks to handle deeper representations, EfficientNet's compound scaling efficiently utilizes additional layers and parameters to capture complex patterns without risking overfitting or vanishing gradients. While ResNet-18 already improved the model's ability to capture more complex features than a traditional 3-layer CNN, EfficientNet further enhances this capability by effectively scaling the model to meet the demands of intricate image data.

Stem Layer

- EfficientNet begins with a **stem layer** that consists of a **3x3 convolution** with a stride of 2, followed by **batch normalization** and **ReLU activation**.
- This layer processes the input image and reduces its spatial dimensions, making it more manageable for deeper layers.

Mobile Inverted Bottleneck Convolution (MBConv) Blocks

The core of EfficientNet is a series of **MBConv blocks**. Each MBConv block is based on **depthwise separable convolutions** and an **inverted residual structure**, adapted from MobileNetV2.

- **Depthwise Convolution**: Each spatial channel is processed separately, reducing computation compared to standard convolutions.
- **Pointwise Convolution (1x1 Convolution)**: After depthwise convolution, a pointwise (1x1) convolution merges the channels back together, allowing for efficient channel-wise interaction.
- **Squeeze-and-Excitation (SE) Module**: EfficientNet includes SE modules, which adaptively recalibrate the importance of each channel, enhancing critical features and reducing noise. This module boosts feature relevance by applying learnable weights.

The MBConv blocks vary in depth, width, and resolution across EfficientNet variants, following the **compound scaling** method.

Swish Activation Function

EfficientNet uses the **Swish** activation function instead of ReLU. The Swish function allows for smoother gradients and slightly improved performance, especially in deeper layers.

Batch Normalization Layers

Batch normalization is applied after each convolution in the MBConv blocks. This normalizes activations, stabilizes training, and reduces internal covariate shift, making it easier to train deeper networks.

Global Average Pooling

After all the MBConv blocks, the feature maps are spatially averaged using **global average pooling** to reduce the dimensions to a single vector per channel. This pooling layer distills the feature map into a global representation of the image, significantly reducing the model's dimensionality before the final classification layer.

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.

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).

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.

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. Batch Size (16 vs. 32 or 64)

The batch size plays a significant role in both the stability of gradient estimates and the memory requirements 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, smaller batch sizes introduce more variability in gradient estimates, which can help the model generalize better by adding noise that allows it to escape local minima.

Due to the constraints of the current environment, I used a batch size of **16** instead of 32 or 64. While this smaller batch size is slower, requiring more iterations to reach comparable performance levels, it introduces beneficial variability into the training process. This variability helps avoid overfitting and improves generalization, which is particularly useful in complex tasks like skin lesion classification.

By comparing the current model with previous versions, I found that using a batch size of 16 resulted in a smoother learning curve with more stable training and validation loss, even though it requires more time per epoch. While larger batch sizes can speed up training, batch size 16 offers more stable progress tracking, making it well-suited for environments with limited computational resources and for achieving robust generalization in challenging classification tasks.

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 and Recall

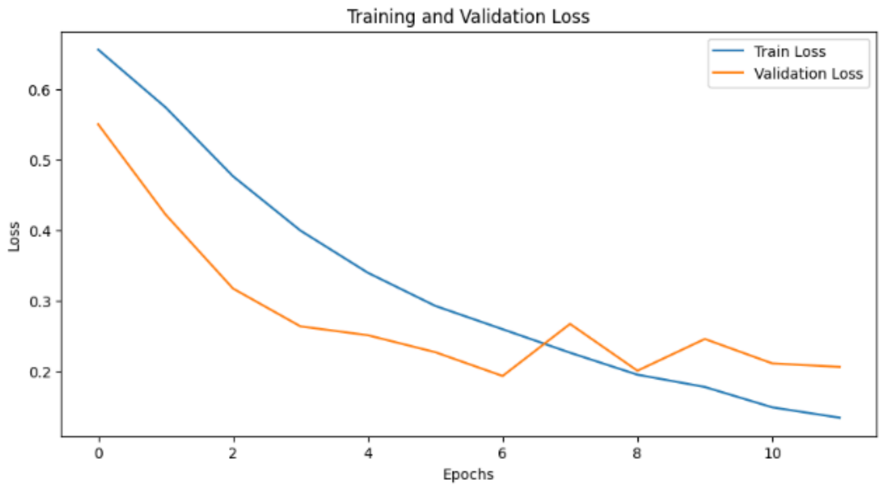
Accuracy represents the percentage of correct predictions out of the total predictions made by the model. It’s calculated as the ratio of correctly predicted instances (both benign and malignant) to the total number of instances. While accuracy provides a straightforward measure of how often the model is getting predictions right, it can be misleading in cases of class imbalance (e.g., if benign cases far outnumber malignant ones). As a result, while accuracy is tracked, it is evaluated alongside other metrics.

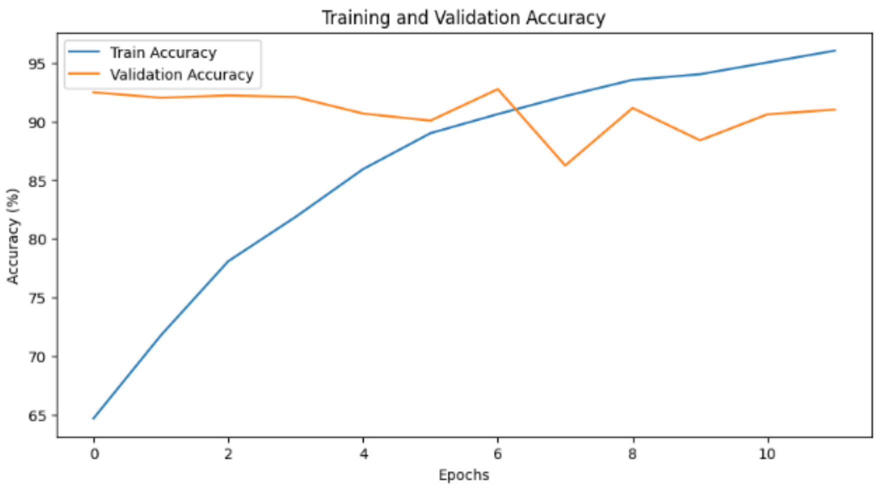
Given the importance of detecting malignant cases, **recall** is a primary focus for this classification task. Recall measures the model's ability to correctly identify malignant instances, calculated as the ratio of true positives to the sum of true positives and false negatives. High recall is essential in this context, as it reflects the model’s ability to catch as many malignant cases as possible, reducing the chance of missed detections.

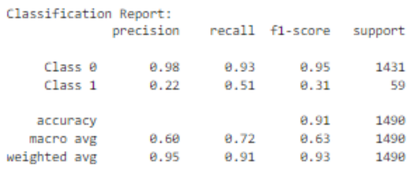
By monitoring recall along with accuracy and ROC AUC, I ensure that the model is effectively capturing the malignant cases rather than merely predicting the majority class. This focus on recall aligns the model’s performance more closely with the goals of the classification task, where identifying malignant lesions is a top priority.

Model Variations

Model 1





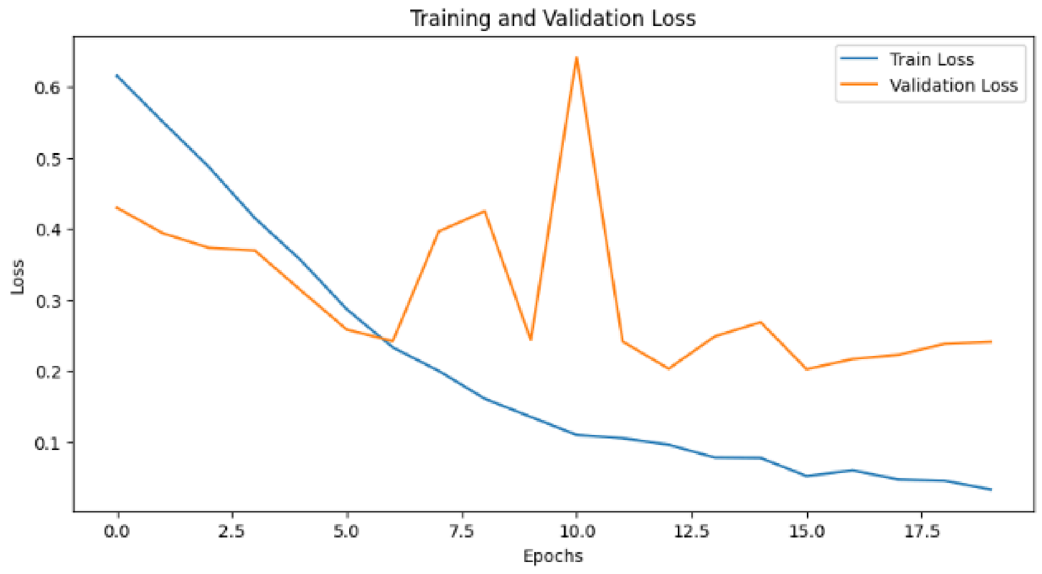


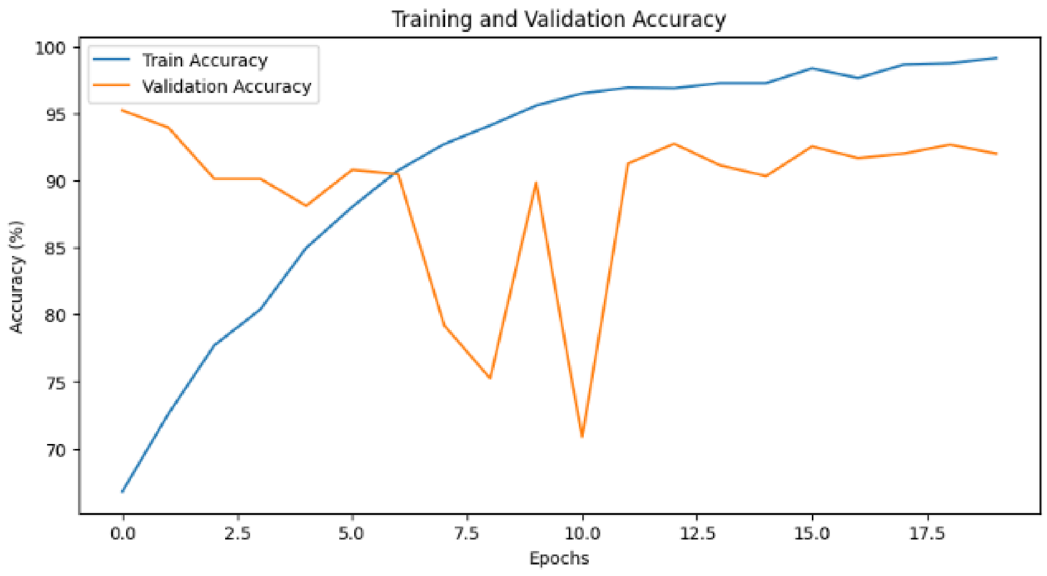
```
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.98      0.93      0.95      1431
     Class 1       0.22      0.51      0.31        59

    accuracy                           0.91      1490
   macro avg       0.60      0.72      0.63      1490
weighted avg       0.95      0.91      0.93      1490
```

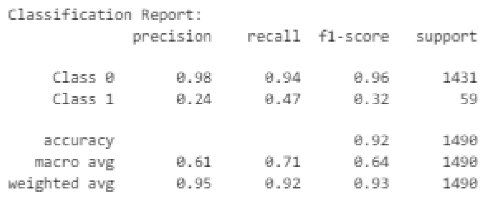
Hyperparameter

- Optimizer Adam with learning rate of 1.1621608010269284e-05 (used optuna to search for best lr)
- Binary Cross Entropy Loss function
- 20 epochs
- Batch size of 16

Model 2





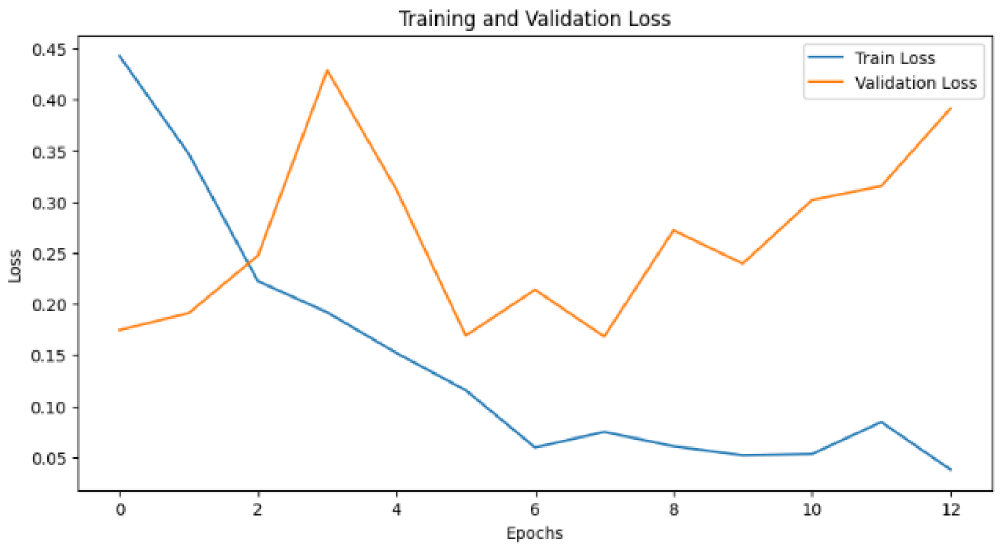


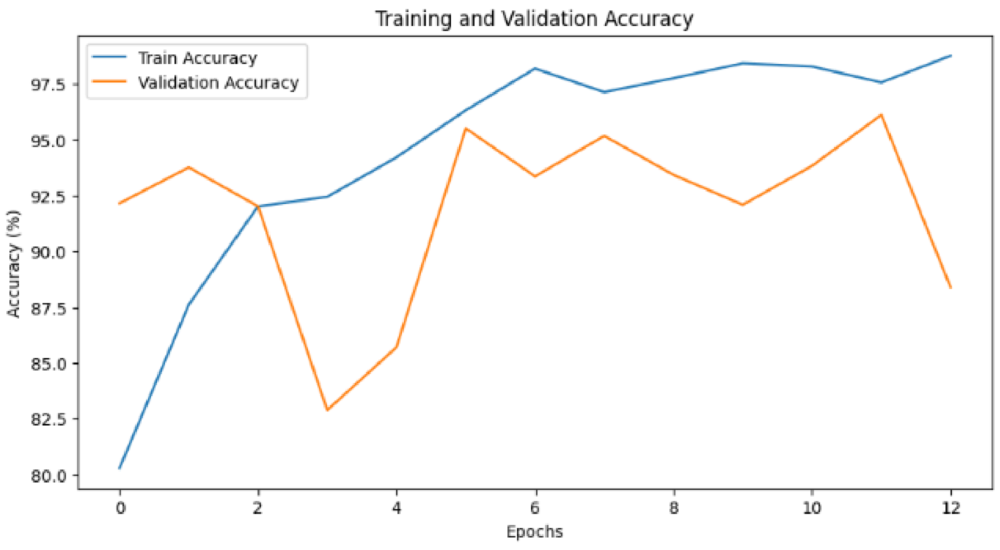
```
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.98      0.94      0.96      1431
     Class 1       0.24      0.47      0.32        59

    accuracy                           0.92      1490
   macro avg       0.61      0.71      0.64      1490
weighted avg       0.95      0.92      0.93      1490
```

Hyperparameter

- Optimizer Stochastic gradient descent with learning rate of 0.01
- Binary Cross Entropy Loss function
- 20 epochs
- Batch size of 16

Model 3





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

     Class 0       0.99      0.89      0.94      1431
     Class 1       0.22      0.76      0.34        59

    accuracy                           0.88      1490
   macro avg       0.60      0.83      0.64      1490
weighted avg       0.96      0.88      0.91      1490
```

Hyperparameter

- Optimizer Adam with learning rate of 0.001
- Binary Cross Entropy Loss function
- 20 epochs
- Batch size of 16

Variation	Recall	Loss	pAUC-aboveTPR
Model 1	0.51	Train Loss: 0.2592 Val Loss: 0.1929	0.1050
Model 2	0.47	Train Loss: 0.0518 Val Loss: 0.2024	0.1228
Model 3	0.76	Train Loss: 0.0595 Val Loss: 0.1682	0.1443

Model Variation Analysis

I observed that Model 1's training and validation losses demonstrate a more consistent decline compared to Model 2 and Model 3, likely due to the use of Adam, which may have contributed to more stable and effective learning. In contrast, the loss curves for Model 2 and Model 3 appear less stable, potentially due to adjustments in hyperparameters or learning rate that introduced more variability into the learning process.

Given the dataset's imbalance, relying solely on accuracy can be misleading, as high precision in the majority class does not necessarily translate to strong performance on the minority class. Therefore, metrics that better capture performance on the minority class are prioritized. The competition organizer’s chosen metric, pAUC-aboveTPR, offers a more accurate measure of the model's effectiveness in identifying minority class instances. Based on pAUC-aboveTPR, Model 3 outperforms both Model 1 and Model 2 with a score of 0.1443, making it the best-performing model for this task.

Additionally, I prioritize recall, as it reflects the model’s ability to correctly identify minority class instances. Model 3 achieves the highest recall at 76%, outperforming the other models in capturing minority class cases. This makes Model 3 particularly valuable for tasks where identifying all relevant instances is crucial, despite its less stable loss curves.

Conclusion

In conclusion, I have selected Model 3 as the final model for this week. The training and validation loss graphs, along with the accuracy graph, indicate promising improvements as the epochs progress. More importantly, Model 3 outperforms the other models in terms of the key performance metrics, pAUC-aboveTPR and recall, which are essential for this task. The emphasis of pAUC-aboveTPR on sensitivity and control over false positives makes Model 3 the optimal choice.

The classification report further highlights Model 3's effectiveness, with a recall of 76% for Class 1, demonstrating its ability to identify minority class instances more accurately than the other models. Although the precision for Class 1 decreased, the boost in recall provides better coverage for critical cases, aligning with the priorities of this task.

Additionally, this week's model architecture has shown marked improvements over last week's, with significant advancements in overall performance and effectiveness. These enhancements confirm Model 3 as the most suitable model for addressing the challenges posed by this dataset.