Melanoma Classification Using CNN and EfficientNet

Group Rantanen | Brandon Cheung, Kareem Hamideh, Karim Grar, Kyungdon Baik
School of Information, The University of Texas at Austin
I 320D: Machine Learning Project
Dr. Mishra
Dec 11, 2025

Introduction

Our project focuses on building a machine learning system capable of identifying melanoma from dermatologic images. Melanoma becomes significantly more dangerous when it is detected late, but it is highly treatable when caught early. Since many people first notice changes in their skin outside a clinical setting, an image-based machine learning tool has the potential to support early screening and make the triage process safer and more confident. Our goal is to create an image classifier that determines whether a skin lesion is malignant or benign using a labeled dataset of melanoma images.

We worked with the Melanoma Cancer Image Dataset from Kaggle, which contains thousands of prelabeled images. The images came in different formats, so we prepared them by resizing, normalizing pixel values, and building TensorFlow datasets that used caching, shuffling, batching, and prefetching to improve efficiency. Because this task requires recognizing detailed visual patterns, deep learning models are the most suitable approach.

To explore different modeling strategies, we trained three models. The first was a custom Convolutional Neural Network that we built from scratch. We experimented with activation functions, batch normalization, dropout, and data augmentation to improve generalization. We then moved to transfer learning by training EfficientNetB0 in two stages. In the first stage, the base layers were frozen while the new classification layers learned the task. In the second stage, we unfroze deeper layers and trained with a smaller learning rate. This two-phase fine-tuning process helped the model gradually adapt to melanoma images. Finally, we repeated the workflow with EfficientNetB3, a larger model that learns more detailed visual features.

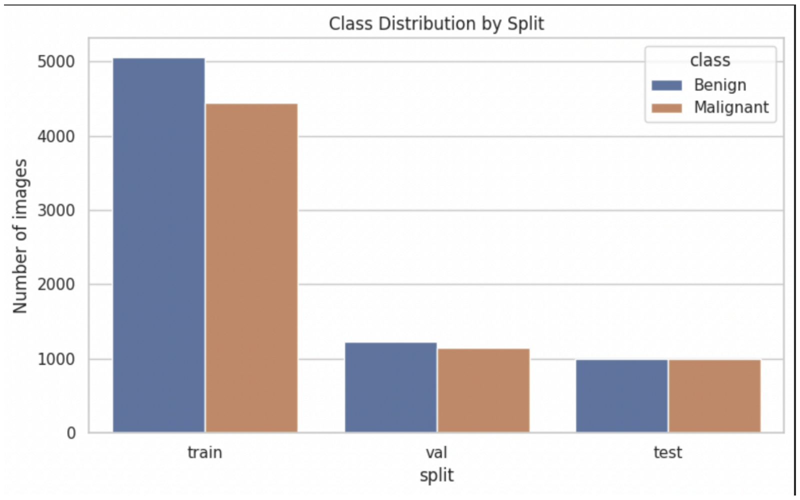
The goal of the project is to compare these approaches, understand where each model performs well or struggles, and evaluate how reliably they can classify melanoma images. We analyze accuracy, AUC, confusion matrices, ROC curves, and examples where the models make consistent errors. By the end, we want to determine which model is most reliable and how factors such as model size, augmentation, and fine-tuning affect performance on this medical image task.

Data Description

The dataset used in this project is the Melanoma Cancer Image Dataset available on Kaggle. It contains thousands of dermatologic images labeled as either benign or malignant. Each image represents a skin lesion photographed under different lighting conditions, angles, and resolutions. The dataset is roughly 83 megabytes in size, and the image dimensions vary, which made preprocessing necessary before training.

To prepare the data, we organized the images into training, validation, and test sets. Each image conveniently came in uniform size of 224 by 224 pixels. We resized the image for the

EfficientNet B3 model which allowed all models to process the data reliably. Pixel values were normalized to the range [0, 1] only for CNN model, since consistent scaling helps neural networks learn more effectively. We then created TensorFlow datasets and applied caching, shuffling, batching, and prefetching to improve loading speed and reduce bottlenecks during training. Additionally, our validation and test splits were pretty balanced and had similar amounts of benign and malignant data. However, our train split was slightly skewed towards benign and could potentially introduce a minor bias in our model's performance.



Data augmentation techniques were added to improve generalization and reduce overfitting. These included random flipping, rotation, zooming, contrast adjustments, and translation. These transformations introduce minor variations to the images during training, helping the models learn more robust features and preventing them from memorizing the training set. Rotating and flipping were used to account for the fact that lesions have no correct orientation. Zooming and translating were used because lesions vary in size and central focus. At the same time, contrast adjustment was applied because the dataset indicates that the images were captured under different lighting conditions.

Overall, the dataset provides a realistic and challenging set of medical images. The variation in color, shape, and texture across the lesions makes it well-suited for testing how effectively different models can identify melanoma types in real-world scenarios.

Method

To approach the melanoma classification task, we tested three different deep learning models. All models were trained using TensorFlow and Keras, and we kept the same data pipeline for consistency. This allowed us to compare the impact of model architecture and fine-tuning methods on performance.

We started with a custom Convolutional Neural Network that we built from the ground up. The model used several blocks of convolutional layers followed by batch normalization and max pooling. After the convolutional blocks, we added dense layers for classification along with dropout to reduce overfitting. ReLU was used as the primary activation function. Data augmentation was applied directly within the model, so each training image was randomly transformed during training. These transformations helped the model learn general visual patterns rather than memorize the training data.

Next, we moved to transfer learning with EfficientNetB0. We trained this model in two stages. In the first stage, the base EfficientNetB0 layers were frozen, leaving only the newly added classification layers to be updated. This allowed the model to learn how to use the pretrained ImageNet features for melanoma classification. In the second stage, we unfroze the deeper layers and continued training with a smaller learning rate. This two-phase fine-tuning process helped the model adapt gradually to the melanoma dataset. We also experimented with learning rate changes, label smoothing, weight decay, and different augmentation strengths to stabilize learning and improve accuracy.

Finally, we applied the same transfer learning approach to EfficientNetB3, which is a larger and more powerful model. Because it requires higher resolution input, the image size was set to 300 by 300 pixels, and the batch size was adjusted to fit the model in memory. We again trained the model in two phases, first with the base layers frozen and then with the deeper layers unfrozen. We chose this two phase setup because it allows the new classification layers to adapt safely before updating the pretrained weights, which research has shown leads to more stable and accurate training in medical imaging. EfficientNetB3 required substantially longer training times, but it consistently produced stronger results than the other models.

Across all models, we used callbacks such as early stopping, model checkpoints, and learning rate scheduling. Early stopping prevented overtraining by restoring the best weights when validation performance stopped improving. Model checkpoints allowed us to save the best version of each model during training. The learning rate scheduler helped stabilize training during the fine-tuning phases. Together, these techniques improved training efficiency and consistency across all three models.

Efficient Net Architecture

EfficientNet is a family of convolutional neural networks designed to scale in a balanced and systematic way. Traditional CNN architectures often improve performance by increasing either the depth, width, or input resolution of the model. Still, they do so unevenly, which leads to diminishing returns and inefficient training. EfficientNet introduces a compound scaling method that increases all three dimensions together in a controlled ratio. This design enables larger EfficientNet models to learn more expressive features without wasting computational power. At the core of EfficientNet is the MBConv block, an inverted bottleneck structure originally introduced in MobileNetV2. These blocks compress and then expand feature maps, preserving detail while reducing unnecessary computations. They also incorporate squeeze-and-excitation units, which help the model focus on the most critical channels within each feature map. This combination of ideas allows EfficientNet to achieve significantly stronger accuracy than older CNNs while using fewer parameters.

EfficientNetB0 is the smallest and lightest version of the family. It uses 224×224 input images and serves as the baseline from which all other models are scaled. Despite being the least significant variant, B0 already has sufficient depth and structure to perform well on a challenging task such as melanoma classification. By leveraging ImageNet pretrained weights, the model begins training with a strong understanding of edges, colors, textures, and shape patterns. These foundational features carry over well to dermatologic images, which often require the model to distinguish subtle changes in color gradients or irregular borders. During our training, B0 adapted quickly once the base layers were unfrozen, and learning rate scheduling helped stabilize its performance.

EfficientNetB3 is a scaled-up version that uses more MBConv blocks, wider channels, and a higher input resolution of 300 by 300 pixels. This increase in resolution is significant for melanoma images because many malignant lesions differ from benign lesions through subtle visual cues. Higher resolution allows the model to examine these variations more closely and form more accurate internal representations. Although B3 takes longer to train and requires more memory, the architectural improvements translate directly into stronger performance. During fine-tuning, B3 extracted richer image patterns, which explains why it ultimately achieved the highest accuracy and the strongest ROC curves. In general, the EfficientNet family is well-suited for medical imaging tasks because it balances expressiveness with computational efficiency, and our results strongly reflected this advantage.

Explaining the Code

The initial dataset came with a train and test split, and we further divided the training portion into 80 percent training and 20 percent validation. For image sizes, we used 224 by 224 pixels for the CNN and EfficientNetB0 models and 300 by 300 pixels for EfficientNetB3. We set a random seed for reproducibility. Because the raw dataset listed all malignant images first and benign images later, we enabled shuffling so the validation set would not end up containing only one class.

To make the data pipeline efficient, we used cache and prefetch with AUTOTUNE so the GPU would not stall waiting for images. Only the training set was shuffled again so each batch would contain a good mix of examples. At this point we also defined the augmentation pipeline. The random flips, rotations, zooms, contrast changes, and translations help the model handle the natural variation found in dermatology images.

We then built and compiled our models. For the CNN, we applied augmentation first, normalized pixel values to a 0 to 1 range, and created four convolutional blocks followed by dropout at 0.5 to reduce overfitting. The model was compiled using a learning rate of 1e minus 3 and the Adam optimizer. Adam adjusts the update size automatically while training, which helps the model learn more smoothly and quickly than using a fixed learning rate.

For EfficientNetB0 and B3, we loaded the pretrained backbones and froze all layers in the first phase. We applied our augmentation pipeline and the correct preprocessing function and included dropout before training the new classification head with a learning rate of 1e minus 4. We also used label smoothing to keep the model from becoming overly confident.

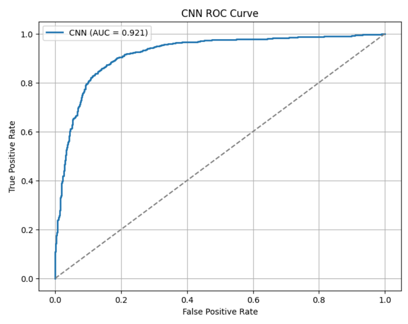
Across all models, we used early stopping to stop training when validation AUC did not improve after five epochs. ReduceLROnPlateau was applied only to the CNN to lower the learning rate when progress slowed. EfficientNet models used cosine decay instead, which gradually lowers the learning rate along a smooth curve so training starts with larger updates and ends with smaller, more precise ones.

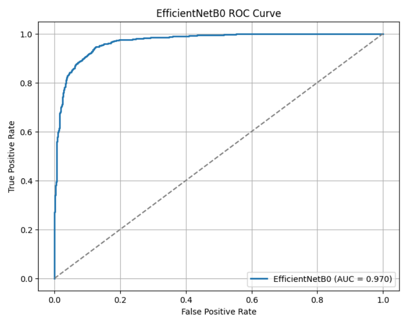
The CNN trained for twenty epochs with early stopping and learning rate reduction. For EfficientNetB0 and B3, we used two phase fine tuning based on the approach recommended by Tajbakhsh et al. 2016 for medical imaging tasks. Phase one trained only the new classification layers. In phase two, we unfroze the backbone except for the batch normalization layers to avoid destabilizing pretrained weights. We compiled the model again using cosine decay for the learning rate and AdamW as the optimizer. AdamW includes weight decay, which helps prevent overfitting during fine tuning. Label smoothing was applied again in this phase.

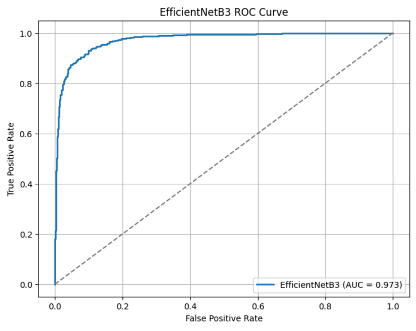
Results

To evaluate our models, we compared the performance of the custom CNN, EfficientNetB0, and EfficientNetB3 on the test dataset. Each model was trained with the same preprocessing and augmentation pipeline to offer a fair and consistent comparison.

The custom CNN served as our baseline, and training took 42 minutes. Early versions of the CNN achieved accuracies below 80%. Still, after experimenting with the number of training parameters, we improved the final accuracy of 85%. The CNN's confusion matrix showed that it struggled most with malignant cases, as expected for a simpler model with fewer parameters. The ROC curve and AUC score reinforced this observation, showing weaker class separation compared to the transfer learning models. While performance improved with architectural tuning, the CNN consistently underperformed EfficientNetB0 and EfficientNetB3 across decision thresholds, reflecting its limited capacity to reliably distinguish malignant lesions.

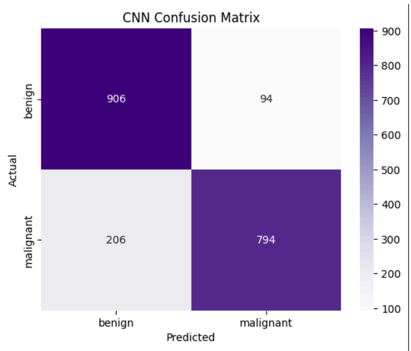


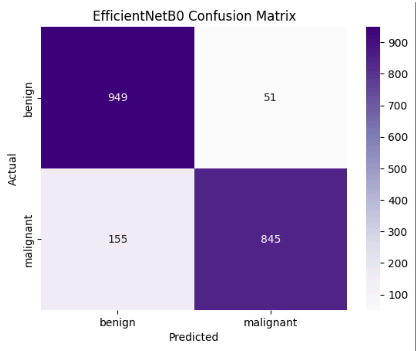


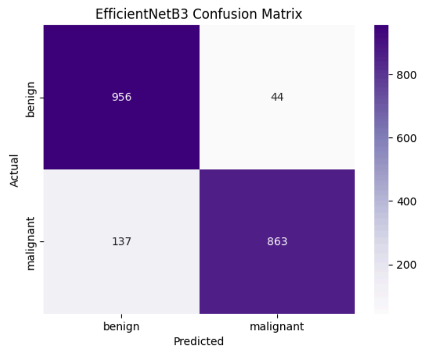


EfficientNetB0 produced significantly better results and was trained for 10 hours. During the first training stage, with the base layers frozen, the model was trained for the whole 20 epochs and established a strong starting point. In the second stage, we unfroze all layers except BatchNormalization layers and trained the model with a smaller learning rate using cosine decay and weight decay. This fine-tuned model achieved test accuracy of 89.7%. The ROC curve and AUC demonstrated a clear improvement in class discrimination over the baseline CNN, with EfficientNetB0 maintaining stronger separation between benign and malignant cases across a wide range of thresholds. This improved discriminative ability was reflected in the confusion matrix, which showed a more balanced tradeoff between true positives and true negatives.

EfficientNetB3 delivered the strongest performance of all models and was trained for 14 hours. Because it is a larger architecture, training required higher resolution images and smaller batch sizes, and fine-tuning took significantly longer. The frozen stage ran for 14 epochs, and the fine-tuning stage continued for 15 more epochs before stopping early. The final accuracy reached 90.95% on the test set. EfficientNetB3 demonstrated the strongest overall discriminative performance, achieving the highest AUC and maintaining superior separation between benign and malignant lesions across nearly all decision thresholds.







cnn

epoch	accuracy	auc	loss	val_accuracy	val_auc	val_loss	learning_rate
1	0.7125	0.7192	24.1621	0.6333	0.628	148.9903	0.001
2	0.737	0.7452	21.3897	0.7133	0.7519	4.3164	0.001
3	0.7495	0.7783	5.7403	0.7347	0.7905	1.8468	0.001
4	0.7566	0.808	2.1155	0.7655	0.8694	1.0351	0.001
5	0.7656	0.823	1.6458	0.8101	0.8712	0.9934	0.001
6	0.7667	0.833	1.3673	0.8366	0.9011	0.6955	0.001
7	0.7718	0.8366	1.3805	0.8295	0.8867	0.9743	0.001
8	0.7659	0.826	1.6021	0.7406	0.8721	0.9828	0.001
9	0.7655	0.8237	1.6985	0.7975	0.8703	1.2404	0.001
10	0.7924	0.86	0.9472	0.8244	0.9022	0.5584	0.0003
11	0.8026	0.8716	0.6506	0.7937	0.8555	0.8162	0.0003
12	0.7961	0.8665	0.6893	0.8295	0.9097	0.4777	0.0003
13	0.8003	0.8738	0.6335	0.8362	0.9127	0.5509	0.0003
14	0.7957	0.8719	0.67	0.8337	0.9111	0.4768	0.0003
15	0.7941	0.8622	0.8104	0.8463	0.9156	0.4786	0.0003
16	0.7996	0.8669	0.7451	0.8295	0.8993	0.6069	0.0003
17	0.8029	0.8744	0.6947	0.8021	0.8959	0.7551	0.0003
18	0.8204	0.8972	0.4906	0.8434	0.9204	0.4062	9E-05
19	0.8263	0.9045	0.4324	0.8429	0.9212	0.3759	9E-05
20	0.8238	0.8973	0.4467	0.8434	0.9175	0.3898	9E-05

effnetb0_frozen

epoch	accuracy	auc	loss	val_accuracy	val_auc	val_loss
1	0.7657	0.8509	0.5015	0.8223	0.9213	0.4206
2	0.8344	0.9117	0.4171	0.8189	0.9269	0.4267
3	0.8435	0.92	0.4043	0.8451	0.931	0.3986
4	0.8442	0.925	0.396	0.8429	0.9333	0.3949
5	0.8575	0.9312	0.3837	0.8564	0.936	0.382
6	0.856	0.9316	0.3829	0.8514	0.9368	0.3868
7	0.8564	0.9338	0.38	0.8594	0.9373	0.3753
8	0.8581	0.9349	0.377	0.8535	0.9377	0.3813
9	0.8641	0.9374	0.3721	0.8581	0.9396	0.3752
10	0.8641	0.937	0.3725	0.8547	0.9406	0.3801
11	0.8668	0.9391	0.368	0.8463	0.9406	0.3885
12	0.8643	0.9381	0.3708	0.8488	0.9402	0.3865
13	0.8683	0.9416	0.363	0.8543	0.9389	0.3825
14	0.8693	0.9408	0.3651	0.8518	0.9415	0.3799
15	0.8727	0.9434	0.359	0.848	0.941	0.3883
16	0.8691	0.9413	0.3649	0.8526	0.9411	0.38
17	0.8688	0.9418	0.363	0.8594	0.942	0.3704
18	0.8705	0.9431	0.36	0.8442	0.9415	0.3953
19	0.8692	0.9426	0.3632	0.8547	0.9426	0.375
20	0.8731	0.9429	0.3606	0.8539	0.9423	0.3776

effnetb0_finetune

epoch	accuracy	auc	loss	val_accuracy	val_auc	val_loss
1	0.8751	0.9443	0.3653	0.8741	0.9486	0.3593
2	0.8824	0.9492	0.3548	0.8577	0.95	0.3911
3	0.8848	0.9505	0.3527	0.8796	0.9539	0.3483
4	0.8846	0.9535	0.3443	0.8804	0.9562	0.3572
5	0.8934	0.9578	0.3336	0.8792	0.9559	0.3491
6	0.8941	0.959	0.3302	0.8884	0.9585	0.3429
7	0.8997	0.9608	0.3248	0.8884	0.9618	0.3332
8	0.9017	0.964	0.3156	0.8897	0.9604	0.3306
9	0.9027	0.9656	0.3106	0.8901	0.9609	0.3339
10	0.9037	0.9652	0.313	0.8956	0.9627	0.3169
11	0.9125	0.9684	0.3006	0.904	0.9639	0.3164
12	0.9126	0.9693	0.299	0.9002	0.9631	0.3228
13	0.9091	0.969	0.3013	0.9027	0.9643	0.3216
14	0.914	0.9719	0.2921	0.8981	0.9643	0.3346
15	0.917	0.971	0.2943	0.9086	0.9657	0.3142
16	0.9159	0.9726	0.2901	0.9099	0.9645	0.3148
17	0.915	0.9712	0.2928	0.9032	0.9651	0.3223
18	0.9184	0.9724	0.2899	0.9069	0.9647	0.3172
19	0.919	0.973	0.2874	0.9069	0.965	0.3173
20	0.9208	0.9746	0.2833	0.912	0.9657	0.3161

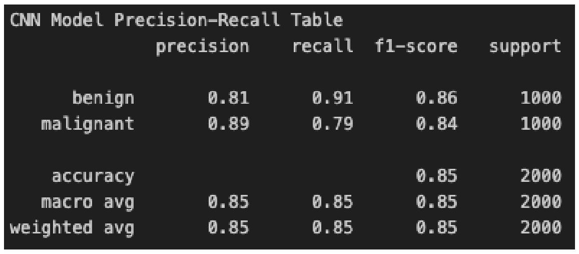
effnetb3_frozen

epoch	accuracy	auc	loss	val_accuracy	val_auc	val_loss
1	0.8124	0.8992	0.4379	0.8573	0.9313	0.389
2	0.8495	0.9243	0.3962	0.8627	0.9346	0.3817
3	0.857	0.9301	0.3851	0.8661	0.9387	0.369
4	0.8586	0.9322	0.3806	0.8653	0.9401	0.3804
5	0.8655	0.9379	0.3696	0.8657	0.9393	0.3692
6	0.8631	0.9358	0.3738	0.8661	0.9421	0.3641
7	0.8628	0.9385	0.3686	0.8707	0.9436	0.3599
8	0.8668	0.9396	0.3655	0.8686	0.9429	0.3652
9	0.8656	0.9385	0.3683	0.8703	0.9444	0.3575
10	0.8675	0.9411	0.3636	0.8712	0.943	0.3613
11	0.873	0.9435	0.3573	0.8745	0.9438	0.3613
12	0.869	0.9412	0.3629	0.8703	0.9439	0.3591
13	0.8732	0.945	0.355	0.8699	0.9424	0.3614
14	0.8723	0.9431	0.3595	0.8712	0.9424	0.3621

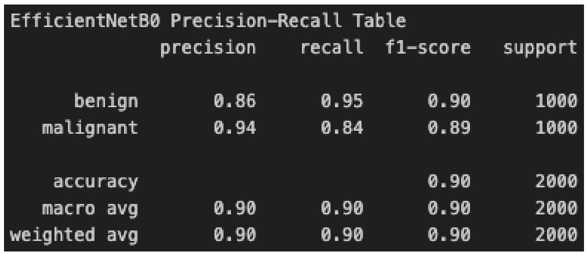
effnetb3_finetune

epoch	train_accuracy	train_auc	train_loss	val_accuracy	val_auc	val_loss
1	0.9129	0.9701	0.2931	0.9015	0.9672	0.3324
2	0.9148	0.9716	0.2884	0.9011	0.9652	0.3238
3	0.9189	0.9752	0.2775	0.8935	0.9648	0.3514
4	0.9254	0.9778	0.2697	0.9158	0.9672	0.3007
5	0.9289	0.9778	0.2669	0.9133	0.9664	0.3043
6	0.9339	0.9802	0.2572	0.9099	0.9682	0.3193
7	0.935	0.9822	0.2502	0.9217	0.969	0.2986
8	0.9372	0.9828	0.2489	0.9141	0.9681	0.2994
9	0.9426	0.984	0.2413	0.9175	0.9678	0.3039
10	0.9437	0.9849	0.2382	0.9158	0.9682	0.3017
11	0.9454	0.9871	0.231	0.9213	0.9678	0.3024
12	0.9492	0.988	0.2283	0.9175	0.9681	0.3054
13	0.9522	0.9892	0.2222	0.9196	0.9684	0.3007
14	0.9543	0.9906	0.2159	0.9208	0.9685	0.3011
15	0.9499	0.9884	0.2249	0.9213	0.9674	0.3032

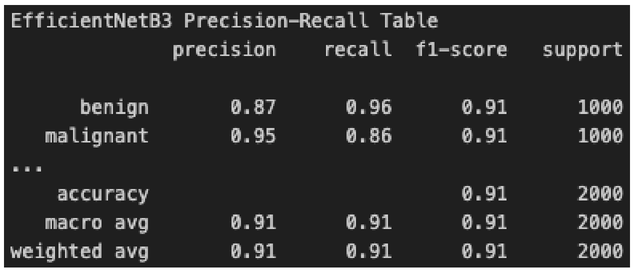
Training metrics showed stable and consistent convergence across all models. The custom CNN improved gradually but achieved lower validation accuracy and AUC than the transfer learning approaches. In contrast, EfficientNetB0 and EfficientNetB3 began with substantially higher validation AUC values, reflecting the benefit of pretrained features. Two-phase fine-tuning further improved performance for both EfficientNet models, with validation accuracy and AUC increasing without a significant gap from training metrics, indicating good generalization. EfficientNetB3 achieved the highest overall validation performance, consistent with its increased capacity and higher input resolution.



```
CNN Model Precision-Recall Table
              precision    recall  f1-score   support

      benign       0.81      0.91      0.86      1000
   malignant       0.89      0.79      0.84      1000

    accuracy                           0.85      2000
   macro avg       0.85      0.85      0.85      2000
weighted avg       0.85      0.85      0.85      2000
```



```
EfficientNetB0 Precision-Recall Table
              precision    recall  f1-score   support

      benign       0.86      0.95      0.90      1000
   malignant       0.94      0.84      0.89      1000

    accuracy                           0.90      2000
   macro avg       0.90      0.90      0.90      2000
weighted avg       0.90      0.90      0.90      2000
```



```
EfficientNetB3 Precision-Recall Table
              precision    recall  f1-score   support

      benign       0.87      0.96      0.91      1000
   malignant       0.95      0.86      0.91      1000
...
    accuracy                           0.91      2000
   macro avg       0.91      0.91      0.91      2000
weighted avg       0.91      0.91      0.91      2000
```

We also created precision and recall tables for all three models. The precision–recall tables showed a consistent improvement across models. Compared to the CNN, EfficientNetB0 and EfficientNetB3 achieved higher recall for malignant lesions while maintaining strong precision, resulting in fewer false negatives and improved overall classification balance.

Overall, the results show a clear trend. The custom CNN provides a solid baseline, EfficientNetB0 improves performance through transfer learning and fine-tuning, and EfficientNetB3 delivers the strongest results through a deeper architecture and more expressive feature extraction. Each model benefits from careful tuning of learning rate schedules, augmentation strategies, and training configurations, and even small parameter changes produce meaningful differences in performance.

Conclusion, Limitations, and Future Work

In conclusion, this project demonstrates how different deep learning architectures approach melanoma classification and how model complexity influences overall performance. The custom CNN established a workable baseline and showed that a model trained entirely from scratch can learn useful visual features. Still, it lacked the representational capacity needed for reliable medical predictions. EfficientNetB0 improved substantially through transfer learning, adapting pretrained ImageNet features to the melanoma dataset, enabling it to capture more detailed texture and color variations. EfficientNetB3 performed the strongest, benefiting from higher input resolution, more expressive feature extraction, and a deeper architecture. The improvements between models were consistent across accuracy, AUC, confusion matrices, and recall for the malignant class, which is especially important for medical applications. These results highlight the value of modern architectures and fine-tuning strategies in building dependable image classification systems. Even though B3 offered the highest accuracy, the resources required to train and deploy it are significantly higher. As a result, EfficientNetB0 provides the best balance between performance and efficiency. It delivers strong accuracy while remaining small enough to train and run on modest hardware, making it a more realistic choice for real-world systems where energy usage, cost, and latency matter.

While the models performed well, several limitations affected the project. The dataset exhibited class imbalance between benign and malignant images, which may have made it more difficult for the models—particularly the CNN trained from scratch—to learn the minority class. Resizing images to a fixed resolution removed subtle details that dermatologists rely on for melanoma diagnosis, and these details could be necessary for model performance. Additionally, although EfficientNet models improved accuracy, they still misclassified some malignant cases, posing a significant risk in medical settings. Training was also computationally expensive for EfficientNetB3, and the project was constrained by available hardware. Another limitation is the dataset's self-similarity. Since the images all came from one Kaggle source, the models may not generalize well to new clinics, devices, or lighting conditions. These practical limitations reinforce the case for a middle-ground model like EfficientNetB0 in situations where reliability must be balanced with time, energy consumption, and deployment constraints.

These limitations create several opportunities for future work. One approach would be to integrate techniques that address class imbalance, such as class weighting, oversampling, or focal loss, to help reduce false negatives. Another improvement would be to train models on higher-resolution images, since melanoma features often depend on fine-grained detail, and EfficientNetB4 or B5 could benefit from such inputs. Exploring Vision Transformers may also be beneficial, as they have recently shown strong performance in medical imaging tasks. Incorporating explainability tools like Grad-CAM could help interpret model predictions and make the system more transparent for clinical use. Expanding the dataset to include images from diverse sources and imaging conditions would also improve generalization. Finally, evaluating runtime performance, energy usage, and the feasibility of on-device deployment would help determine which model is best suited for practical screening tools.

Q&A

1. Can you explain your full augmentation pipeline and why each transformation was chosen?

For augmentation, we used flips, rotations, zoom, translations, and contrast changes. Each one was chosen because dermatology images naturally vary in those ways. Lesions don't have a fixed orientation, so flips and rotations help the model stop relying on direction. Zoom and small shifts make the model less sensitive to how centered or close the lesion is in the frame. Contrast changes help the model deal with different lighting conditions across the dataset. The point wasn't to distort the images but to expose the model to the kinds of variation it would actually see, so it learns the features of the lesion rather than memorizing the exact training images.

2. Why did you use two-phase fine-tuning, and what benefits did you expect?

We used two-phase fine-tuning because it's a well-supported approach in transfer learning research. We read the paper "Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?" that specifically recommended training the new classification layers first while keeping the pretrained base frozen, then unfreezing deeper layers and continuing with a smaller learning rate. The idea is to avoid disrupting the pretrained ImageNet features too early. The paper showed that this staged method leads to more stable training and better accuracy, especially in medical imaging, so we followed that strategy for EfficientNetB0 and B3.

3. What is EfficientNet, and what architectural principles does it rely on?

EfficientNet scales depth, width, and resolution together so the model grows in a balanced and efficient way. It uses MBConv blocks and squeeze-and-excitation units to focus on important features without wasting computation. B0 is the lightweight baseline, while B3 is a higher-resolution version that captures finer details and performs better on melanoma images.

4. Can you explain cosine learning rate decay in simple conceptual terms?

Cosine decay essentially starts the learning rate high then slowly drops following a smooth curve shaped like a half cosine wave. Early on, the model takes bigger steps so it can learn quickly, and later the steps naturally get smaller so it doesn't overshoot or destabilize.

5. Why didn't you superimpose ROC curves for direct model comparison?

We didn't put all the ROC curves on one plot because they were almost on top of each other, which made the graph messy and hard to tell apart. When curves overlap like that, you can't really see the differences anyway, so it's easier and clearer to show each model's ROC curve separately.

6. What are the dataset's exact statistics, and how does imbalance affect training?

The dataset had 6,289 benign images and 5,590 malignant images, so benign cases were slightly more common in our splits. The imbalance is not severe, but even a small imbalance could potentially influence how a model behaves. It is something we still need to examine more closely, especially since we suspect the model may lean slightly toward predicting the more common class. Understanding how this imbalance affects confidence and error patterns would help clarify whether it had a meaningful impact on our results.

7. What steps did you take to ensure efficient training during 4-fold (or multi-fold) validation?

We did not use multifold validation.

8. What was your training time for each model, and what optimizations mattered most?

Training time depended heavily on the model size. The CNN finished in about 40 minutes, while EfficientNetB0 took around 10 hours and EfficientNetB3 took closer to 14. What mattered most for keeping these times reasonable was using a fast input pipeline, early stopping so we didn't overtrain, and the two-phase fine-tuning setup, which kept the larger models stable instead of wasting time on bad learning rates or unstable updates.

Code Repository:

https://github.com/kmano3/melanoma_classifier

References

https://www.kaggle.com/datasets/bhaveshmittal/melanoma-cancer-dataset/data

https://www.mayoclinic.org/diseases-conditions/melanoma/symptoms-causes/syc-20374884

https://huggingface.co/google/efficientnet-b0

https://huggingface.co/google/efficientnet-b3

https://www.ibm.com/think/topics/convolutional-neural-networks

https://www.geeksforgeeks.org/computer-vision/efficientnet-architecture/

https://www.geeksforgeeks.org/machine-learning/image-classifier-using-cnn/

https://ieeexplore.ieee.org/document/7780684

https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/

https://pubmed.ncbi.nlm.nih.gov/26978662/

https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf