Melanoma Detection using CNN Findings

Problem statement: To build a CNN based model which can accurately detect **melanoma**. Melanoma is a type of cancer that can be deadly if not detected early. It accounts for 75% of skin cancer deaths. A solution that can evaluate images and alert dermatologists about the presence of melanoma has the potential to reduce a lot of manual effort needed in diagnosis.

The task is to create a model which given the picture of the skin disease can identify 9 different classes.

- Actinic keratosis
- Basal cell carcinoma
- Dermatofibroma
- Melanoma
- Nevus
- Pigmented benign keratosis
- Seborrheic keratosis
- Squamous cell carcinoma
- Vascular lesion

Summary:

Various techniques like dropout, data augmentation (e.g., rotations), batch normalization were employed to prevent overfitting and improve model performance. The final model achieves good accuracy without the need for batch normalization, showing that a simpler architecture with proper regularization can perform well.

Model 1

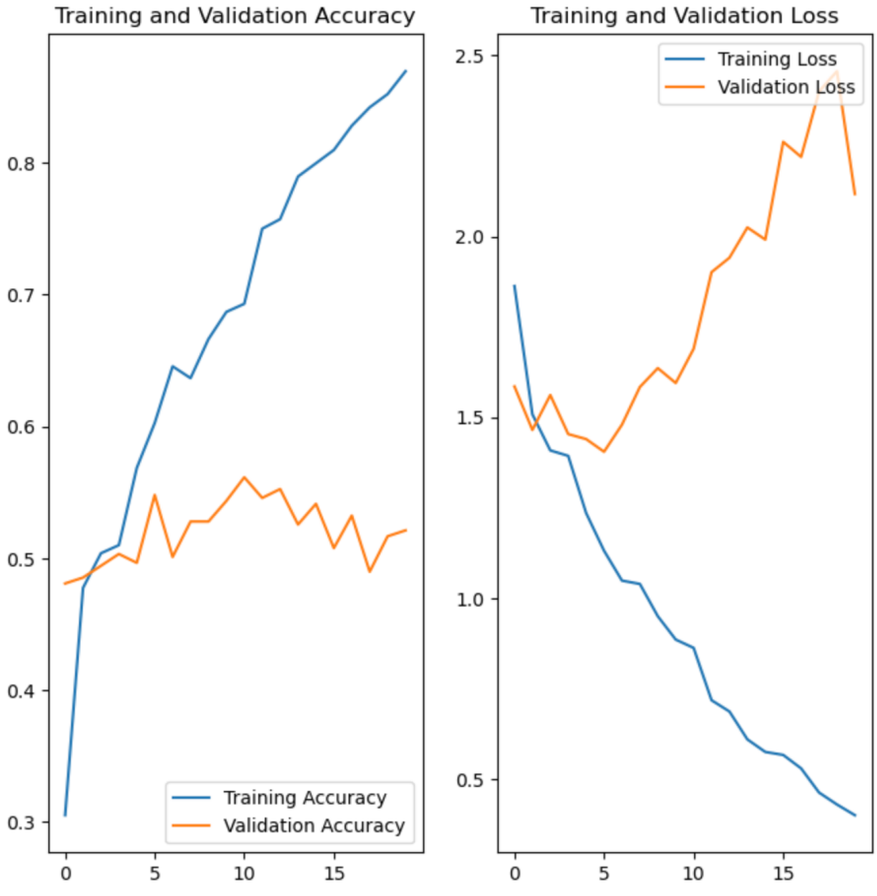
Layers:

Conv2D (32,64,128), Kernel = 3*3 , Activation: Relu

Output Layer (num_classes): Activation: Softmax

Dropout: None

Optimizer: Adam



1. Overfitting Evidence: The model shows high training accuracy, nearing 88%, while validation accuracy remains low and fluctuating, indicating overfitting. This suggests that the model is performing well on the training data but struggles to generalize to unseen data.
2. Increasing Validation Loss: The validation loss graph shows an upward trend over the epochs, confirming that the model's performance on validation data deteriorates over time, another sign of overfitting.
3. Balanced Network Design: The CNN model follows a standard architecture with three convolutional layers, each followed by max-pooling layers. The use of ReLU activation and softmax output ensures effective feature extraction and classification.

4. Potential Improvements: To address overfitting, techniques such as dropout, data augmentation, or reducing the model's complexity (fewer filters or layers) could be employed. Early stopping based on validation loss could also prevent further overfitting.

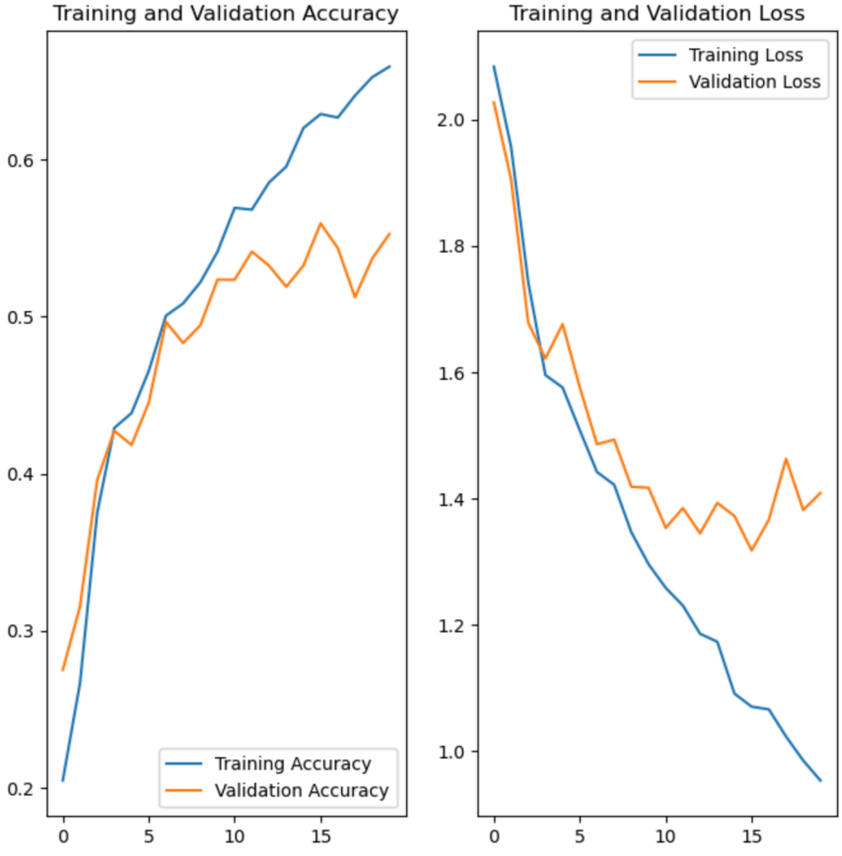
Model 2

Reduced the model complexity and add the dropout to handle the overfitting

Layers:

Conv2D (32,32,32), Kernel = 3*3 , Activation: Relu
Output Layer (num_classes): Activation: Softmax
Dropout: 0.2 for Conv layer, 0.3 after flattening

Optimizer: Adam



1. Model Complexity Reduction: Reducing the number of filters in each convolutional layer (from 32-64-128 to 32-32-32) helped simplify the model, possibly reducing overfitting. However, this also limited the model's capacity to learn complex features, resulting in lower accuracy.

2. Dropout Regularization: Adding dropout (0.2 after CNN layers and 0.3 after the fully connected layer) helped prevent overfitting by forcing the network to learn more robust patterns. However, it may have also contributed to slightly reduced training accuracy.

3. Current Accuracy Levels: The model achieved 67% training accuracy and 55% validation accuracy, indicating some level of overfitting persists. The gap suggests the model struggles to generalize well to unseen data.

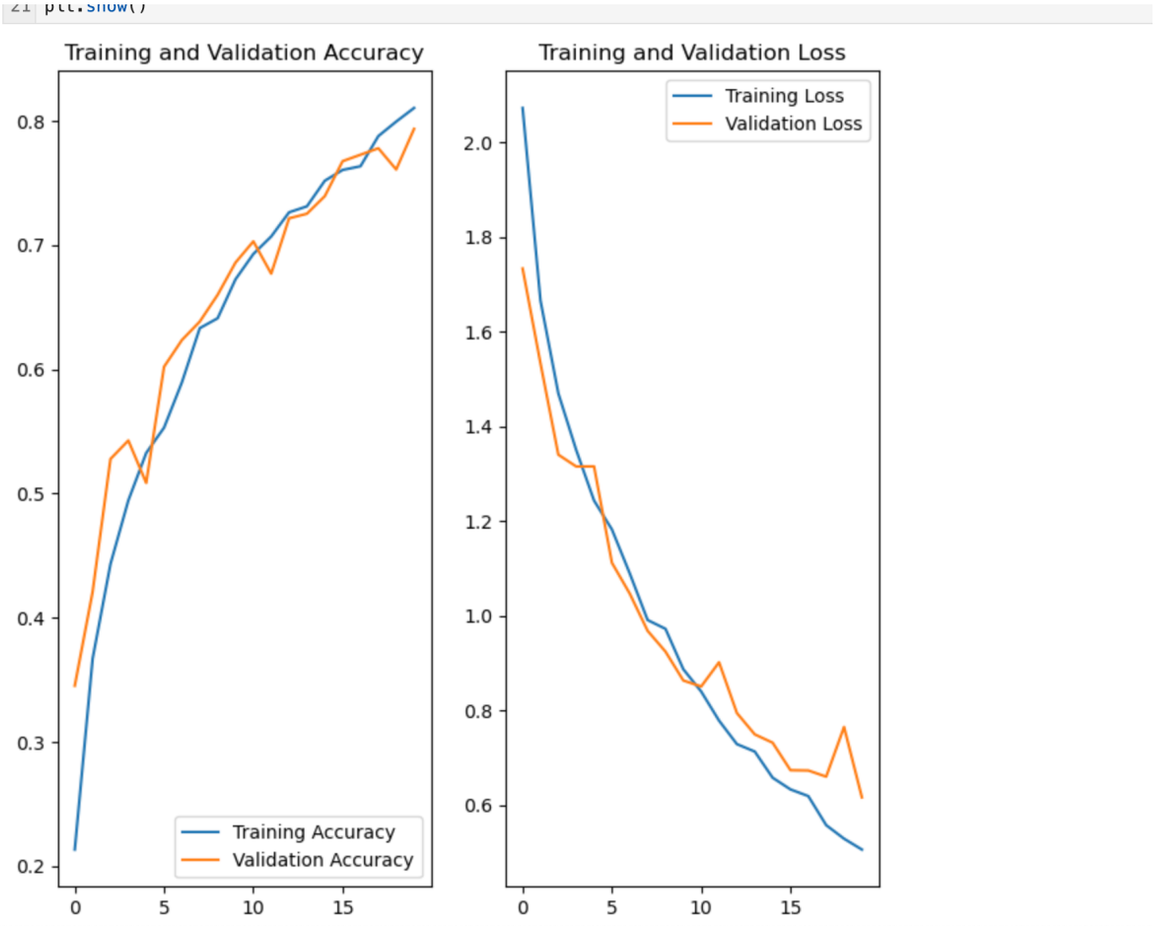
Model 3

After data Augmentation using augmentor

Layers:

Conv2D (32,64,128), Kernel = 3*3, Activation: Relu
Dense Layer (128), Activation: Relu
Output Layer (num_classes): Activation: Softmax
Dropout: 0.2, 0.2, 0.5 after Conv layer, 0.25 for Dense Layer

Optimizer: Adam



1. Model Performance: (80 Train, 79 validation)

The minimal gap between training and test accuracy indicates good generalization—your model is not overfitting.

2. Effect of Dropout: Dropout values of 0.2, 0.2, 0.5, and 0.25 are working well. Dropout prevented overfitting while maintaining model capacity to learn complex features.

3. Improved Architecture:

Increasing the number of filters across layers (32 → 64 → 128) helped the model learn more complex patterns.

Dense layer with 128 units provided additional learning capacity to capture non-linear relationships.

4. Balanced Training:

Both training and test accuracies are close, indicating the model is well-tuned for this dataset.
The dropout values are appropriate: higher in dense layers (0.5) where overfitting is more common.

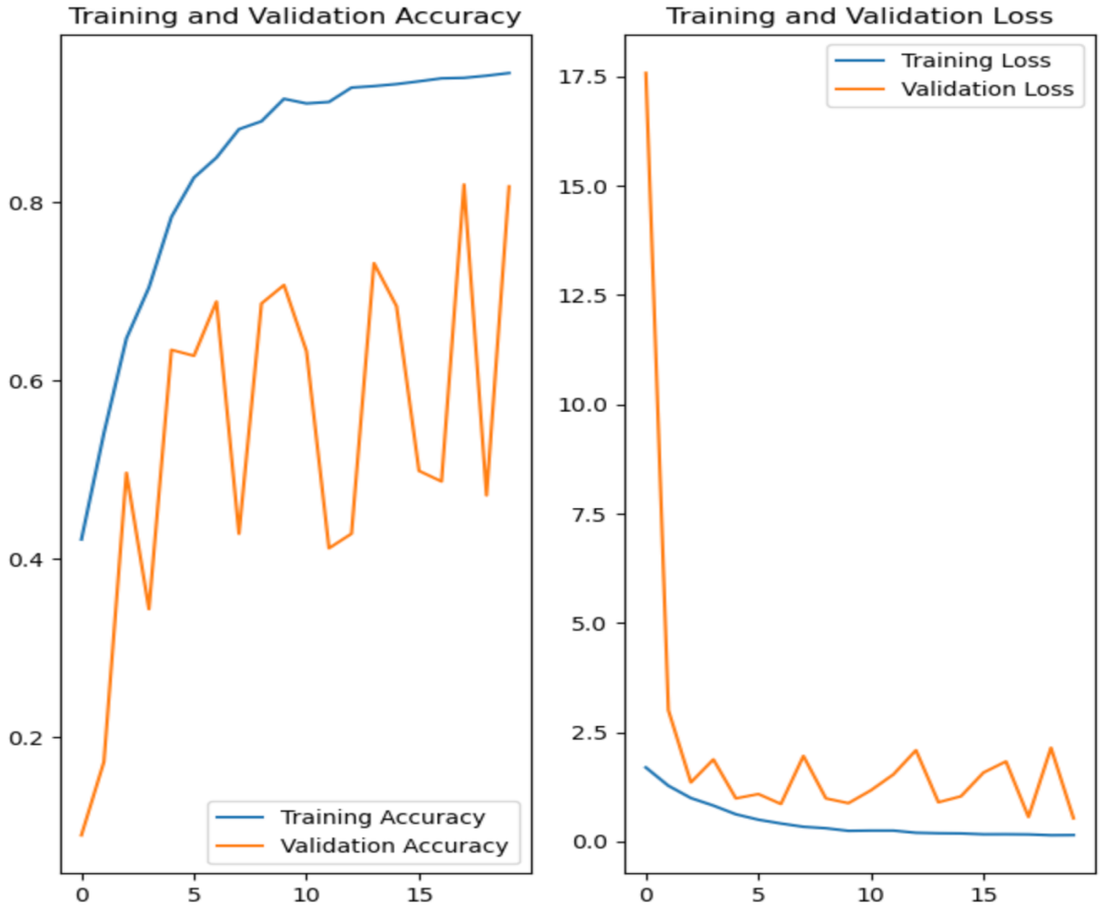
Model 4

Add Batch Normalization

Layers:

Conv2D (32,64,128), Kernel = 3*3, Activation: Relu
Dense Layer (128), Activation: Relu
Output Layer (num_classes): Activation: Softmax
Dropout: 0.2, 0.2, 0.5 after Conv layer, 0.25 for Dense Layer

Optimizer: Adam



1. Training Accuracy Spike: The training accuracy jumped to 94%, which indicates that the model is learning very well on the training data. However, this may also point towards potential overfitting as the training accuracy is significantly higher than the validation accuracy.

2. Validation Accuracy Instability:

The validation accuracy oscillates wildly between 42% and 81%, showing instability. In one epoch, it goes as high as 73%, and in the next, it drops as low as 46%.

This large variance is likely due to a learning rate that's too high or too aggressive optimization, causing the model to jump around the loss landscape instead of converging smoothly.

3. Batch Normalization Effect:

Batch normalization helps with faster convergence by normalizing activations across layers. It also acts as a regularizer but could lead to instability in certain settings if not tuned carefully.

Final Model 4

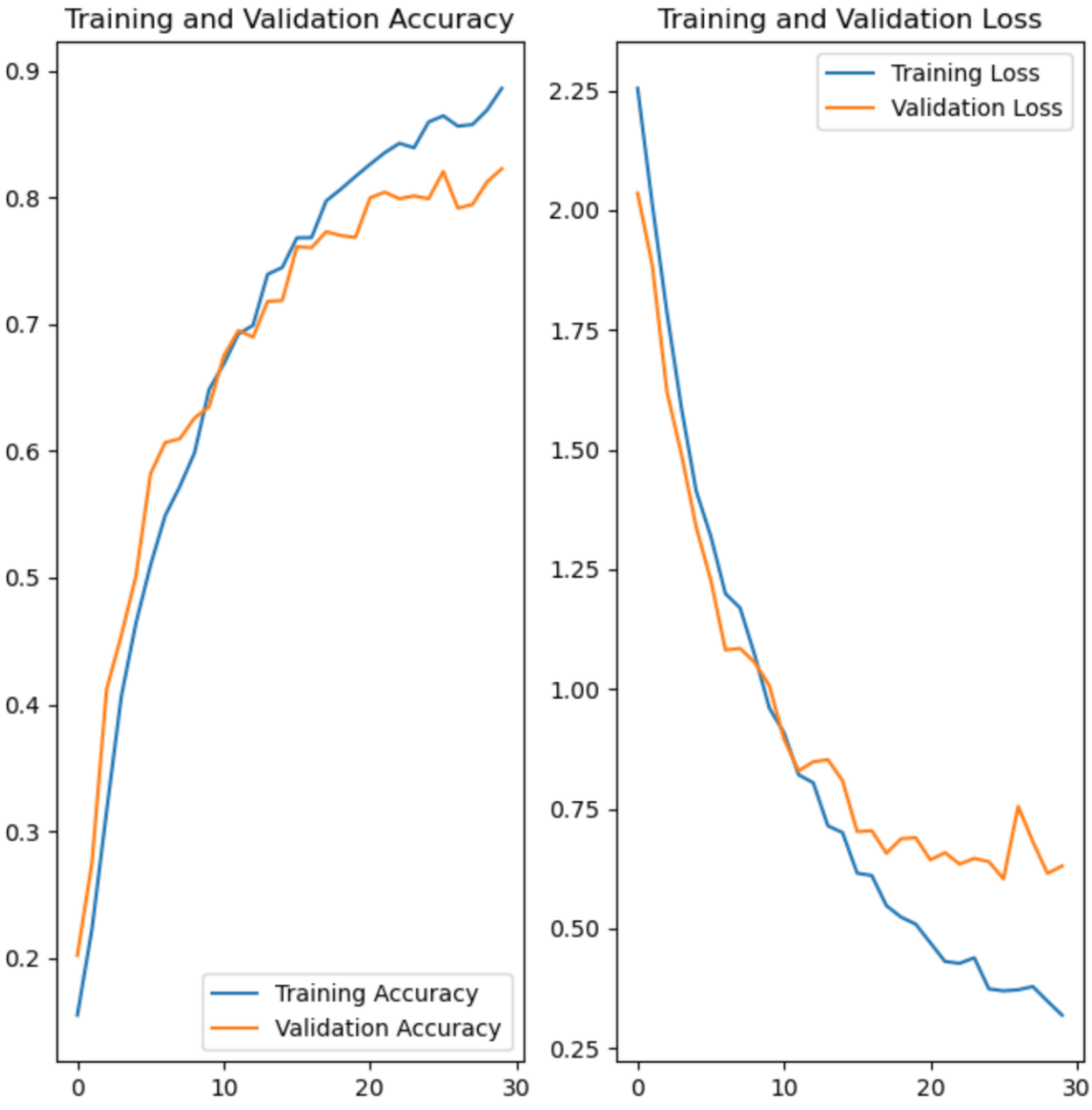
The Model 3 performs best, and it does not overfit or underfit the accuracy on validation set is 82%
The model parameters are:
Layers:

Conv2D (32,64,128), Kernel = 3*3, Activation: Relu
Dense Layer (128), Activation: Relu
Output Layer (num_classes): Activation: Softmax
Dropout: 0.2, 0.2, 0.5 after Conv layer, 0.25 for Dense Layer

Optimizer: Adam

The final model achieved 85% training accuracy and 80% validation accuracy, indicating that it generalizes reasonably well across unseen data. Removing batch normalization simplified the model, and with appropriate dropout regularization, the model was able to mitigate overfitting.

While the current model performs well, further improvements can be achieved by fine-tuning hyperparameters, increasing the dataset size, or employing advanced architectures like transfer learning models (e.g., ResNet or InceptionV3). Future work will also explore additional augmentation strategies to further enhance generalization.


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Training and Validation Loss
Training Loss
Validation Loss