School of Arts and Sciences

Department of Computer Science and Mathematics

Melanoma Cancer Predictive Model

Tamim Eter – 202206700 - 81920221

Abdul Rahman Al Zaatari – 202201380 – 81906611

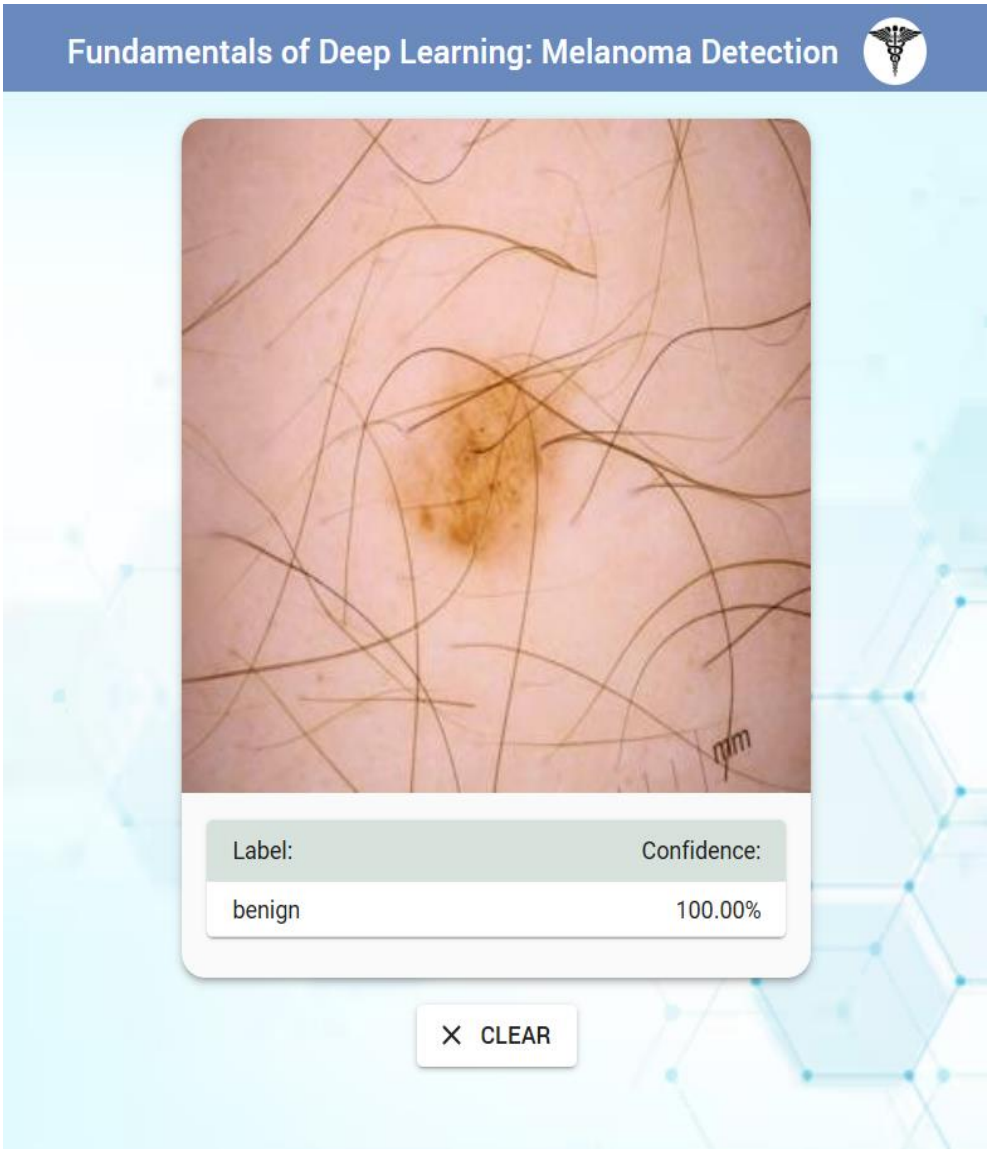
CSC 461: Fundamentals of deep learning

Dr.Seifedine Kadry

November 27th, 2024

Deployed Project

Project was deployed in 3 methods:
Deployed on Render
Deployed on AWS
Deployed Locally using Flask



1- Project Proposal:

Goal and Scope:

Our project aims to develop an image recognition system using deep learning algorithms to accurately predict melanoma skin cancer from skin lesion images. Melanoma is a deadly form of skin cancer that affects millions annually, and early detection can save countless lives.

The scope of this project includes developing a robust model capable of distinguishing malignant melanoma from benign skin conditions. The model will leverage high-quality clinical datasets and advanced image classification techniques to provide reliable predictions. Detecting melanoma typically requires specialized dermatological expertise and advanced diagnostic tools, which often makes early detection inaccessible for many people around the world. By utilizing deep learning algorithms, we aim to bridge this gap and offer an early, cost-effective diagnostic solution. This approach has the potential to improve treatment outcomes and significantly increase survival rates.

Dataset Name: Melanoma Skin Cancer Dataset (10,000 Images)

Source:
The dataset includes high-quality skin lesion images, categorized into two classes: **malignant** and **benign**. It is publicly available under a **CC0: Public Domain license**, allowing unrestricted use for research and development.

Structure and Size:

- **Total Images:** 10,000
    - **Training Set:** 9,600 images
    - **Evaluation Set:** 1,000 images
- **Classes:**
    - Malignant (cancerous)
    - Benign (non-cancerous)

Key Features:

- Each image captures detailed skin lesion characteristics necessary for accurate classification.
- Balanced distribution between malignant and benign samples ensures fair training conditions.

Usability:
The dataset has a usability score of **7.50**, indicating it is well-structured and suitable for building deep learning models for melanoma classification.

Potential challenges: Training a deep learning model on a large dataset demands significant computational resources, particularly GPUs. To address this, we will utilize cloud platforms such as **Google Collab**, which offer access to powerful GPUs for efficient model training.

2 - Data Preparation and Preprocessing:

Dataset Description

- The dataset is already organized into two primary sets:
- **9,600 images** for training
- **1,000 images** for testing
- These images are stored in separate directories for training and testing. However, within the training set (9,600 images), we further split the data to create a **validation set**. Using ImageDataGenerator with the validation_split parameter, we allocate 20% of the training data for validation purposes, leaving 80% for actual training. Specifically:
- **80% of training data** (7,680 images) is used for training the model.
- **20% of training data** (1,920 images) is reserved as a validation set to monitor the model's performance during training.
- This structured approach ensures a balanced distribution of data and facilitates both effective training and reliable performance evaluation.

Standardization

- We applied **standardization** using the rescale=1./255 technique from TensorFlow's Keras library to normalize pixel values in the images. This process scales pixel values from the range [0, 255] to [0, 1], ensuring consistency across the dataset. Standardization accelerates model convergence during training and is an essential preprocessing step for deep learning models.

Data Augmentation

To make the model robust against variations in lesion orientation, we implemented **data augmentation techniques**, including:

- **Rotation Augmentation** (rotation_range=20): Randomly rotates images by up to ±20 degrees to simulate different orientations.
- **Width and Height Shifts**: Introduces positional variations in the images.
- **Zooming**: Simulates changes in the camera's zoom level.
- **Horizontal Flipping**: Creates mirrored versions of the images.
- **Validation Split**: Allocates a portion of the data for validation to monitor performance during training.
- These augmentation techniques increase dataset diversity, enhance model generalizability, and reduce the risk of overfitting to specific patterns in the training data.

3 - Model Architecture and Design:

Model Choice: We chose the Convolutional Neural Network (CNN) for our melanoma skin cancer classification task. CNN can automatically learn spatial feature hierarchies from images which makes it suitable and effective for identifying differences between malignant and benign skin lesions. CNNs are effective at detecting key patterns such as edges, textures and shapes which is particularly crucial for medical image classification problems.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
flatten (Flatten)	(None, 186624)	0
dense (Dense)	(None, 64)	11,944,000
dense_1 (Dense)	(None, 1)	65

Total params: 11,963,457 (45.64 MB)
Trainable params: 11,963,457 (45.64 MB)
Non-trainable params: 0 (0.00 B)

Layer Design: In terms of layer design, the architecture of the model consists of several types of layers, each serving a specific purpose in the learning process. The **Convolutional Layer (Conv2D)** is implemented with two configurations. The first applies 32 filters of size 3x3 to the input images, and the second applies 64 filters of the same size. Each filter in the first layer detects features such as edges and textures in the images, while the second layer allows the network to capture more complex features from the input images. The activation function used in both layers is **ReLU**, which introduces non-linearity and helps the model learn complex patterns. The **Pooling Layer (MaxPooling2D)** is applied after each Convolutional Layer with a 2x2 pool size, reducing the spatial dimensions of the feature maps while retaining the most important features. The **Flattening Layer (Flatten)** is used after the Convolutional and Pooling Layers to flatten the feature maps into a 1D vector that can be fed into the fully connected layers. A **Fully Connected Layer (Dense)** with 64 units and ReLU activation is added to learn higher-level features from the extracted patterns. Finally, the output layer is another Dense layer with a single unit and a sigmoid activation function.

Model Complexity: In terms of model complexity, our model is relatively simple, computationally efficient, and suitable for smaller datasets because it consists of only a few Convolutional and Dense layers. Using data augmentation techniques, the model can capture and learn essential patterns in the datasets without overfitting due to its simplicity. In contrast, adding more layers or increasing the number of filters would increase the model's complexity and its ability to learn more intricate patterns. However, this would also increase the training time and computational requirements, potentially leading to overfitting if the dataset is small. This baseline model serves as a strong foundation for image classification tasks, and its performance could be improved by exploring more advanced architectures, such as pre-trained models or deeper CNN designs.

Pretrained Model: We later used the pre-trained **ResNet18** model for this task due to its ability to deliver improved performance, especially with limited datasets. The model benefits from fast convergence and efficient knowledge transfer, as it has already learned to extract general image features from diverse datasets. This pretraining enhances its generalizability on specialized datasets, such as melanoma images. The pretrained ResNet18 comes with an effective set of parameters that can be fine-tuned for our specific classification task, making it particularly advantageous when working with limited medical datasets.

4 - Training Process:

Training Strategy: We employed various training strategies, including **Transfer Learning**, **Data Augmentation**, and **Early Stopping**, to enhance the performance and generalizability of our model. To increase the diversity of the training data and reduce the risk of overfitting, we applied a range of data augmentation techniques such as rotation, width and height shifts, zooming, and horizontal flipping. The **Transfer Learning** technique involved using a pre-trained model, **ResNet18**, which was already trained on a large dataset like ImageNet and fine-tuning it for our specific task of melanoma classification. Additionally, we implemented **Early Stopping** to halt training when the model's validation performance ceased to improve, preventing overfitting and saving computational resources.

Loss function and Optimization: We implemented a **Convolutional Neural Network (CNN)** model and carefully selected the **loss function** and **optimization techniques** to maximize the model's effectiveness. The **loss function** chosen was **Binary Crossentropy**, as it is well-suited for binary classification problems where the model needs to distinguish between two classes: malignant (cancerous) and benign (non-cancerous). For optimization, we used the **Adam Optimizer (Adaptive Moment Estimation)** due to its computational efficiency, low memory requirements, and adaptability to problems with noisy gradients or sparse data, which could be characteristics of our dataset. The initial **learning rate** was set to 0.001, but during training, we utilized a **learning rate scheduler**, specifically **ReduceLROnPlateau**, to dynamically decrease the learning rate when the validation loss stopped improving. This adjustment ensured the model could fine-tune its parameters without overshooting the optimal weights, ultimately improving its performance.

```
# Hyperparameters for grid search
learning_rates = [1e-3, 1e-4]
optimizers = ['adam', 'sgd']
batch_sizes = [16, 32]
```

Hyperparameter Tuning: We applied hyperparameter tuning to optimize our models' performance by adjusting key parameters using grid search, specifically testing two values for learning rates, batch sizes, and optimizer types. The grid search approach tests combinations of selected values for each parameter. This approach allows us to systematically explore the impact of these parameters on model performance and identify the optimal configuration for accurate melanoma classification. We tested different values of learning rate to control the model weights updates during training, different batches sizes to determine the number of training samples processed before the models' internal parameters are updated, and various optimizers to identify the type that best suits our datasets.

Regularization: To prevent overfitting and enhance generalization, we implemented several regularization techniques:

1. **Learning Rate Scheduling:** We used the StepLR scheduler to reduce the learning rate by a factor of 0.1 every 7 epochs, allowing finer weight adjustments as training progressed.
2. **Batch Normalization:** Built-in batch normalization layers in the ResNet18 model stabilized training by normalizing layer inputs, reducing sensitivity to initial weights.
3. **Data Augmentation:** Techniques like resizing, random horizontal flips, and normalization increased dataset diversity, improving model robustness.
4. **Pre-trained Model:** Fine-tuning ResNet18 leveraged learned features from ImageNet, reducing overfitting while generalizing well to melanoma data.

5. **Optimizer Configuration:** The SGD optimizer with momentum stabilized training, leading to efficient convergence.

By combining these strategies, we ensured a robust and generalized melanoma classification model.

5 - Evaluation and Analysis:

Evaluation Metrics: The evaluation metrics for binary classification problems is crucial particularly in cancer detection. **Recall** is the most critical metric as it measures the model's ability to correctly identify positive cases (malignant cases); a low recall can result in false negatives (FN), leading to undiagnosed cancer, delayed treatment, and increased risk. **Specificity** measures the model's ability to correctly identify negative cases (benign cases), ensuring benign cases are not overclassified as malignant, which helps maintain trust in the model for low-risk cases. **Precision** evaluates the quality of positive predictions by minimizing false positives (FP), which are malignant predictions. The **F1-Score** balances precision and recall, making it especially important in situations with class imbalance, ensuring a more comprehensive evaluation of the model's performance.

Model 1:

Confusion Matrix Interpretation for Model 1 (Without Resnet18)

From the provided matrix:

- **Benign (True Label)**:
    - **False Positives (FP)**: 87 cases incorrectly classified as malignant.
    - **True Negatives (TN)**: 423 cases correctly classified as benign.
- **Malignant (True Label)**:
    - **True Positives (TP)**: 466 cases correctly classified as malignant.
    - **False Negatives (FN)**: 48 cases incorrectly classified as benign.

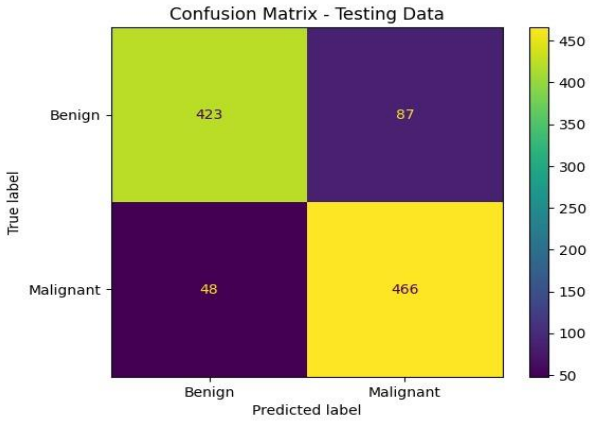


Figure 2: 2 by 2 Confusion Matrix-Testing-CNN

<u>Summary of Results:</u>

Recall:

- Malignant: 0.906
- Benign: 0.831

F1-Score:

- Malignant: 0.874
- Benign: 0.864

We can conclude that the model **is unsuitable** for a healthcare problem, due to its low **Recall** and **F1-score** for malignant cases, **<u>which led us to use Resnet18 for better accuracy.</u>**

Model 2:

Model Architecture

```
Using device: cuda
Classes: ['benign', 'malignant']
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
```

Input Layer: Takes RGB images as input with three channels.

Initial Convolution and Pooling:

- **Conv2D (7x7)**: Extracts low-level features like edges and textures.
- **BatchNorm2D**: Normalizes activations for stable training.
- **ReLU Activation**: Introduces non-linearity.
- **MaxPool2D (3x3)**: Reduces spatial dimensions while retaining key features.

Residual Blocks:

- Organized into four sequential layers (layer1 to layer4), each doubling the number of filters (64 → 128 → 256 → 512) and optionally applying downsampling.
- Each **BasicBlock** consists of:
  - Two **Conv2D (3x3)** layers with BatchNorm and ReLU.
  - Skip connections to prevent vanishing gradients and ease optimization.

Global Average Pooling:

- **AdaptiveAvgPool2D**: Reduces spatial dimensions to 1x1 for feature vector extraction.

Fully Connected Layer:

- **Linear Layer**: Maps 512 input features to 2 output classes (benign and malignant).

Output Layer:

- Applies **Softmax** (via cross-entropy loss) during classification for probability predictions.

Confusion Matrix Interpretation for Model 2 (Resnet18):

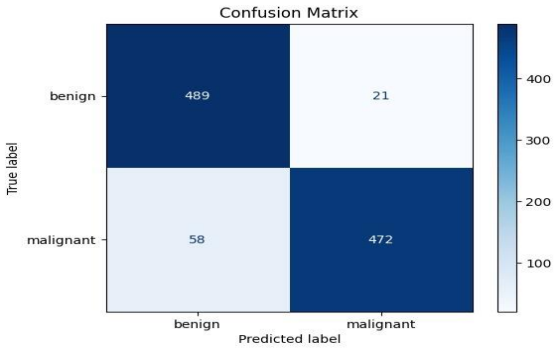


Figure 3: 2 by 2 Confusion Matrix-Testing –ResNet18

Summary of Results:

Recall:

- Malignant: 0.89
- Benign: 0.96

F1-Score:

- Malignant: 0.92
- Benign: 0.92

Model comparison:

Model 1: CNN Without Pretraining

1. **Validation and Training Accuracy:**
   a. Training accuracy started at 59.82% in the first epoch while validation accuracy was equal to 79.07%.
   b. Training accuracy showed a gradual increase, achieving 87.71% by epoch 10.
2. **Generalization and Overfitting:**
   a. Initially, a significant gap existed between training and validation accuracies, indicating overfitting as the model trained on specific patterns from the training data.
   b. Validation loss began at 0.5529 and decreased to 0.3361 by epoch 10, indicating better generalization over time.
3. **Challenges:**
   - The CNN model struggled with generalization, as seen in fluctuating validation accuracy (e.g., 60.18% at epoch 3, 83.19% at epoch 7). Despite some improvement in overall accuracy, the precision, recall, and F1-score metrics highlight significant performance issues.
   - The confusion matrix shows that the CNN model achieved moderate precision (0.89) and recall (0.906 for benign, 0.831 for malignant) when classifying malignant cases. This resulted in a relatively low F1-score of 0.874 for malignant and 0.864 for benign, which indicates suboptimal performance given the medical nature of the dataset. This indicates an inability to effectively distinguish malignant cases, a critical aspect for healthcare applications where false negatives (missed malignant cases) can have severe consequences.

Model 2: ResNet18 (Pretrained)

1. **Performance Overview:** The pretrained ResNet18 achieved a test accuracy of **92%**, with precision, recall, and F1-scores averaging **0.93**, **0.92**, and **0.92**, respectively. Training and validation accuracies aligned closely, reducing overfitting and indicating stable feature learning.
2. **Generalization:** Validation loss stabilized earlier, and the model demonstrated superior generalization to unseen data due to its ability to transfer learned features from diverse datasets.
3. **Error Analysis:** The confusion matrix shows significant improvements in recall for malignant cases (**0.89**) and benign cases (**0.96**), with F1-scores of **0.92** and **0.93**, respectively. This reduced the false negative rate and enhanced classification quality, making ResNet18 highly suitable for medical applications, addressing critical limitations of the initial CNN model.

	precision	recall	f1-score	support
benign	0.89	0.96	0.93	510
malignant	0.96	0.89	0.92	530
accuracy			0.92	1040
macro avg	0.93	0.92	0.92	1040
weighted avg	0.93	0.92	0.92	1040

6 - Conclusion and Future Directions:

Summary of Contributions: In this project, we developed a deep learning-based melanoma skin cancer predictive model. By leveraging advanced convolutional neural networks (CNNs) and the pre-trained ResNet18 model, we effectively addressed the challenge of classifying skin lesions into malignant and benign categories. The project demonstrated significant improvements in performance after applying transfer learning and robust training techniques, achieving over 90% test accuracy with ResNet18. Data augmentation techniques, such as rotation and zoom, enhanced dataset diversity, while early stopping and learning rate schedulers prevented overfitting. These contributions emphasize the potential of deep learning to provide accessible and cost-effective diagnostic tools for melanoma detection.

Limitations:

- Model Complexity: While ResNet18 improved performance, it is relatively simple compared to state-of-the-art architectures such as ResNet50 or EfficientNet. A more complex model might have captured intricate patterns in the data more effectively.

Future Work:

- Incorporate additional datasets with diverse images representing various demographics, lighting conditions, and lesion types to improve generalization.
- Explore synthetic data generation using techniques like Generative Adversarial Networks (GANs) to create realistic, augmented samples.
- Experiment with more sophisticated architectures such as ResNet50, DenseNet, or EfficientNet, which could extract more detailed features and further boost performance.
- Consider ensemble learning to combine the strengths of multiple models.
- Use automated hyperparameter tuning methods, such as Bayesian Optimization, to identify optimal configurations for learning rates, batch sizes, and regularization techniques.