**Deep Learning for Skin Cancer Detection: Classifying Benign and Malignant Skin Lesions Using ResNet34**

Sree Sarada Alapaty

G01440615

HAP 774

Professor: Januz [Wojtusiak](https://canvas.gmu.edu/courses/26245/users/6693)

George Mason University,

Fairfax, VA

**Objective**: The primary focus of this project is to apply deep learning to differentiate skin lesions into benign and malignant. Skin cancers, especially melanoma, are very detrimental to patients’ health and early detection plays a key role in treatment and patient outcomes. Computer systems able to tell benign from malignant skin lesions can assist dermatologists by giving them an early, precise diagnosis.

Using the **Human Against Machine dataset from Harvard Dataverse**, which contains over 10,000 images of various skin lesions, this project focuses on:

* Binary Classification: A distinction between benign (non-cancerous) and malignant (cancerous) tumors, like advanced stages of melanoma, BCC, and actinic keratoses.
* Deep Learning: A trained ResNet-34 model is optimized for this problem. Since the model automatically learns and finds relevant features in images, it is suitable for image classification problems, such as in medical diagnostics.
* Handling Class Imbalance: There are more benign than malignant cases in the dataset, and so these can give us an unfair prediction. Using class weighting, in this project we tackle this problem by making the model more sensitive to malignant cases.
* Enhancing Generalizability: With data augmentation and regularization process, the project will increase the generalizability of the model to invisible data and score higher in validation and test datasets.

By the end of the project, the model should be able to predict whether a particular lesion is benign or malignant, which could be used to identify skin cancer early enough to intervene. This performance metric will provide some indication of how effective the model was at discriminating against these two classes such as accuracy, ROC-AUC, confusion matrix.

**Data Preprocessing and Loading:**

In this step, the path to metadata, images and ground truth files were created. The data set in this project is Human Against Machine dataset from the Harvard Dataverse with 10,000 training images dataset of dermatoscopic skin lesions. We save the images into 2 directories HAM10000\_images\_part\_1 and HAM10000\_images\_part\_2, and we save the ground truth labels of the test set into the ISIC2018\_Task3\_Test\_GroundTruth.csv file.

The following paths were defined:

* **Metadata file**: Contains information about the images such as lesion ID, image ID, diagnosis (dx), diagnosis type (dx\_type), patient age, sex, and localization of the lesion.
* **Image directories**: Two directories containing the training images and one directory for the test images.
* **Ground truth file**: This Contains the actual labels for the test images, which will be used to evaluate the model’s performance.

**Loading Metadata and Preparing Data**: With Pandas, the metadata were read from the CSV file into a DataFrame. These metadata are 10,015 records with an image and patient information. This dataset has columns like:  
  
lesion\_id: Unique identifier for Lesion identifier.  
image\_id: Related image id which locates the image in the dataset.  
dx: Diagnostic value for the lesion (e.g., bkl, nv, mel).  
dx\_type: Diagnostic process (e.g., histopathology).  
age: Patient’s age.  
sex: Patient's sex.  
Region: Area of the body where the lesion was located.

After loading the metadata, a preview print of some initial rows was printed to make sure that the data was loaded correctly and organized. Metadata will be needed to assign labels to images and organize data as input to the machine learning model.

A screenshot of a computer

Description automatically generated

**Loading Ground Truth Data:**

The ground truth labels of the test images are inserted here based on the ISIC2018\_Task3\_Test\_GroundTruth.csv file. This is a data file useful for checking the performance of the model against invisible test data. Diagnoses and other patient specific information like lesion ID, image ID, diagnosis type, age, sex, lesion location etc are stored as the ground truth.  
  
The code transforms the dx diagnostic labels into binary labels for binary classification:  
  
Lesions diagnosed as melanoma (mel), basal cell carcinoma (bcc) or actinic keratosis (akiec) are assigned the number 1 (malignant).  
Every other diagnosis is marked as 0 (normal).

This binary label will be used for both the model training and evaluation, enabling a clear distinction between malignant and benign cases.

A screenshot of a computer

Description automatically generated

**Splitting Data into Training and Validation Sets**: The data was split between training and validation set in 80/20 split, where stratified sampling is done based on the binary labels. This classifying means that the training and validation sets have equal percentage of benign and malignant cases which is very important to avoid class imbalance when model training and evaluation.  
  
The split was done with the train\_test\_split method from sklearn.model\_selection module. A random state was chosen of 42 to be reproducible so that the same split is returned each time the code is executed.  
  
Training set size: 8012 images  
Validation Set Size: 2003 images.

This split provides a large enough training set to build a robust model while reserving a significant portion of the data for validation, enabling reliable model evaluation during training.

A screenshot of a computer program

Description automatically generated

**Sample Image Visualization**: To gain insights into the dataset, a set of sample images from the training set was visualized. The images represent both benign and malignant skin lesions, allowing us to better understand the data the model will be trained on. This visualization helps confirm that the images are correctly labeled and provides a visual understanding of the skin lesions before model training.

A screenshot of a computer screen

Description automatically generated

**Defining a Custom Dataset and Data Loaders**: A custom dataset class, SkinLesionDataset was built to handle the image data of skin lesion effectively. It is a class that takes advantage of torch.utils.data.Dataset and can be used to access images and binary labels for the models during model training.

The \_\_getitem\_\_ method retrieves an image and its label based on the index. The image is loaded using its file path, and the corresponding label (benign or malignant) is fetched from the dataset.

The \_\_len\_\_ method returns the total number of images in the dataset.

This class ensures that each image is properly loaded and processed when passed through the data loaders during training and validation.

**Data Augmentation and Normalization**: Data augmentation methods were used to increase the generality and handling of the model with the dataset changes. These transformations help the model become more robust to changes in image orientation, lighting, and color variations.

The following augmentations were applied:

Resizing: The images are resized to the standard 224x224 pixels.  
Random Horizontal Flip: a few images are randomly flipped horizontally.  
Rotation: Images are randomly rotated up to 20 degrees.  
Color Jitter: Dims brightness, contrast, saturation, and hue slightly to give the images some jerkiness.  
Normalization: The pixels are normalized to a mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225], as would be expected in the case of a pretrained model.

**Creating Data Loaders:** For model training and validation, we built data loaders for the training and validation sets. These loaders will batch and shuffle the data to streamline the training.

Train Loader: Shuffles the training data and loads it in batches of 32 images.

Validation Loader: Loads validation data without shuffling to ensure consistency during evaluation.

A screenshot of a computer program

Description automatically generated

**Loading Pretrained ResNet-34 Model**: For this project, I used a pretrained ResNet-34 model as the base architecture. ResNet-34 - primarily applied to image classifications based on residual connections which helps on reducing the vanishing gradient of deep networks. By using a pretrained model, the network can use the weights that it learned from a large dataset (like ImageNet) to improve the convergence rate and accuracy on smaller datasets like the one used for this project.

The following steps were taken:

Device Setup: The model is loaded to the GPU if available, otherwise it defaults to the CPU.

Model Modification: Since this is a binary classification task (benign vs malignant), the final fully connected (FC) layer of ResNet-34 was modified. The original output layer was replaced with a single output neuron with a linear activation (nn.Linear) for binary classification.

**Model Architecture**:

This architecture shows the layers of the ResNet-34 model, including:

* **Batch normalization** and **convolutional layers** that process image data.
* The final **fully connected layer** (fc), which has been modified for binary classification with one output feature.

A screenshot of a computer code

Description automatically generated

**Defining Class Weights and Loss Function**: Since the class imbalance in the dataset (with much more benign than malignant lesions), class weights were added to the loss function to make sure the model was focusing on the minority class (malignancy). This ensures that the model is not biased towards the majority group.  
  
Class Weights: The benign class (labeled 0) is given a weight of 1.0, the malignant class (labeled 1) is given a weight of 3.0. This weighting reflects the need to correctly diagnosis malignant disease.  
Loss Function: The loss function in binary cross-entropy with logits was BCEWithLogitsLoss for this binary classification task. The class weights were passed as a parameter to ensure proper penalization of errors in the minority class. The use of **BCEWithLogitsLoss** is crucial as it combines the binary cross-entropy loss with a sigmoid activation internally, making it more numerically stable for classification tasks like this.

**Defining the Optimizer and Learning Rate Scheduler**: The loss function was minimised with an Adam optimizer. Here I used a learning rate of 0.001, which is usually the optimal threshold to fine-tune a pretrained model like ResNet-34.  
  
Then a learning rate scheduler was added to enhance training efficiency. The ReduceLROnPlateau scheduler reduces the learning rate as the validation loss plateaus. The following parameters were used:  
  
Factor: Learning rate is reduced by factor of 0.1 when there is a plateau.  
Patience: If validation loss does not change after 3 epochs, the learning rate is adjusted.  
Verbose: True to indicate during training.

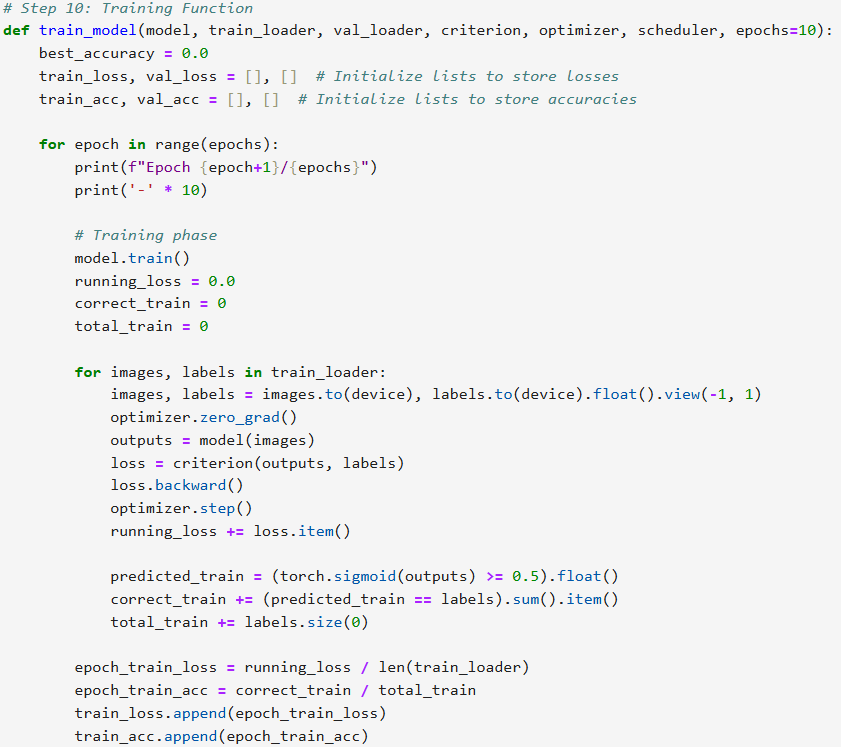
A screenshot of a computer

Description automatically generated

**Defining the Training Function**: The train\_model function is be used to train the ResNet-34 model and observe how the training occurs over several epochs. It’s responsible for training and confirming, adjusting model weights and measuring loss and accuracy.  
  
Training Loop: Every time an epoch, the model is trained in training mode (model.train()). Loss and accuracy are calculated per batch of training set.  
Loss claculation: Binary cross-entropy loss is calculated using BCEWithLogitsLoss function and gradients are backpropagated to update model weights.  
Accuracy calculation: Outputs calculated based on predicted output are set thresholded to 0.5, allowing for images to be labelled as benign or malignant. Correct predictions are tracked to compute the training accuracy.

**Training Metrics**:

* **Loss**: Tracks how well the model is minimizing classification errors over time.
* **Accuracy**: Measures the proportion of correctly classified images per epoch.



**Validation Phase**: The model is tested against the validation set after every training epoch. (model.eval()) in order to not have any gradients added. Validation loss and accuracy are calculated based on model predictions to actual labels in the validation set.  
  
Validation loop: Predictions of the model is matched with the real labels and loss is calculated in the same manner as in the training stage.  
Metrics: Validation loss and accuracy is stored per epoch and printed for monitoring performance.

**Learning Rate Scheduling and Model Saving**: At the end of each epoch, the learning rate is adjusted if validation loss palteaus using ReduceLROnPlateau scheduler. When validation accuracy gets better, it saves the model so the best performing model gets saved.



**Plotting Loss and Accuracy Curves**: Following 10 epochs of training, training and validation loss and accuracy were plotted to observe model performance.  
  
Loss Curve: Has a reduced training and validation loss over time that is a signal the model is learning properly.  
Accuracy Bar: Experiencing a growth in both training and validation accuracy, although validation accuracy is better than training accuracy in the later epochs.  
Such plots tell you what’s been learned by the model and which could be better tuned.

A screenshot of a computer program

Description automatically generatedA screenshot of a graph

Description automatically generated

**Predicting and Displaying Results on Test Images**: The model performance on the individual test images were tested by implementing a function to load the test images, transform and predict. The actual label (benign or malignant) is obtained from the ground truth, and the model’s predicted label is compared.

* Image loading and Preprocessing: Every image is loaded, preprocessed, and runs through the trained model to make predictions.
* Prediction: The model prediction probability is set to 0.5 to label the lesion as benign or malignant.
* Visualization: The image is displayed with its actual and predicted labels.

A screenshot of a computer screen

Description automatically generated

A close-up of a skin disease

Description automatically generated

**Validation Set Evaluation**: The model was evaluated on the validation set, generating a confusion matrix, classification report, and AUC-ROC score.

* **Confusion Matrix**: The matrix shows how well the model classified benign and malignant lesions. The model correctly classified **1182 benign** and **306 malignant** cases, but **430 benign** cases were misclassified as malignant.
* **Classification Report**:
  + **Benign**: High precision (0.93) but lower recall (0.73), indicating some false positives.
  + **Malignant**: Moderate recall (0.78) but lower precision (0.42), reflecting the class imbalance.
  + Overall accuracy: **0.74**
* **AUC-ROC Score**: The **ROC-AUC score of 0.8311** shows that the model has a good ability to distinguish between benign and malignant lesions.

A screenshot of a computer code

Description automatically generated

A blue squares with white text

Description automatically generatedA screenshot of a report

Description automatically generated

**Test Set Performance-Confusion Matrix, Classification Report, and AUC-ROC**: The model was tested on unseen data to assess its generalization performance. The results include a confusion matrix, classification report, and AUC-ROC score for the test set.

* **Confusion Matrix**: The model correctly classified **832 benign** and **265 malignant** cases, but **372 benign** cases were misclassified as malignant.
* **Classification Report**:
  + **Benign**: Precision is high (0.95) but recall is lower (0.69), showing the model is better at correctly predicting benign cases.
  + **Malignant**: Recall (0.86) is relatively strong, but precision is lower (0.41), highlighting the class imbalance issue.
  + Overall accuracy: **0.72**
  + **AUC-ROC Score**: The **ROC-AUC score of 0.8347** indicates that the model performs well at distinguishing between benign and malignant lesions.

A computer screen shot of a program

Description automatically generated

A blue squares with white text

Description automatically generated

**Conclusion & Future Work**: The model performed well on both validation and test sets, having an accuracy of 0.74 in validation and 0.72 in test. With the ROC-AUC of 0.8311 (validation) and 0.8347 (test), the discrimination between benign and malignant lesions is robust. However, the model struggles with precision for malignant cases due to class imbalance, despite using class weighting.

Future Work:

* Address Class Imbalance: Using methods like oversampling, undersampling or Synthetic Minority Over-sampling (SMOTE) to maximize accuracy in malignant cases.
* Try Deeper Architectures: I would like to try deeper models such as ResNet50 or EfficientNet for feature extraction.
* Data Augmentation: Utilizing more powerful data augmentation techniques to boost training variance.
* Hyperparameter Adjustment: Optimize learning rate, batch size, number of epochs and other parameters further.

**References:**

* Dataset:[**https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T**](https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T)
* [**https://www.isic-archive.com/**](https://www.isic-archive.com/)
* **ResNet34:** [**https://pytorch.org/vision/main/models/generated/torchvision.models.resnet34.html**](https://pytorch.org/vision/main/models/generated/torchvision.models.resnet34.html)
* [**https://www.geeksforgeeks.org/compute-classification-report-and-confusion-matrix-in-python/**](https://www.geeksforgeeks.org/compute-classification-report-and-confusion-matrix-in-python/)
* Some help from the ChatGPT when the code ran into errors.