Deep Learning for Finance

Group Project – Melanoma Cancer Detection

Authors:

Luís Ribeiro (nº 20231536)

Renato Morais (nº 20231135)

Fernando Tiago (nº 20231535)

Thiago Bellas (nº 20231131)

Saad Islam (nº 20230513)

Contents

Introduction

Skin cancer poses a significant public health concern globally, with incidence rates steadily rising. Timely detection and accurate diagnosis are pivotal in improving patient outcomes and reducing mortality rates associated with this condition. To address this challenge, we propose the development of a deep learning system tailored for skin cancer detection through image analysis.

The primary objective of this project is to develop a deep learning system for image classification, specifically aimed at detecting benign and malignant tumors from medical images. The project utilizes convolutional neural networks (CNNs), a powerful class of deep learning models, to address this classification task. Through the analysis of dermatoscopic images, the system aims to distinguish between benign and malignant skin lesions, providing valuable assistance to dermatologists in diagnosing skin cancer.

The utilization of CNNs enables the system to learn intricate patterns and features indicative of malignancy directly from the image data. Through comprehensive training on a diverse dataset of dermatoscopic images, the system endeavors to discern subtle visual cues associated with malignant lesions, thereby facilitating accurate and efficient classification.

The significance of this project lies in its potential to enhance early detection and diagnosis of skin cancer, improving patient outcomes. By providing dermatologists with an advanced tool for rapid and reliable screening, our deep learning-based system has the potential to expedite the diagnostic process and ensure timely intervention for patients at risk of skin cancer.

*With this report it was delivered 2 notebooks, one that contains the model that we consider our best one and has the predictions for our dataset ("g4_project_final") and one that contains all the experiments that we have made to achieve a better model ("*project_attempts*"). Through this report we will refer all the attempts we have made in this assignment.*

Task definition

The main goal of this project is to develop a classification model using a Convolutional Neural Network (CNN) to distinguish between benign and malignant melanoma tumors in skin images. The model will be trained on a dataset consisting of 13,879 images, where each image contains information about the presence or absence of melanoma. The task involves creating a model capable of analyzing these images and providing an accurate prediction about the nature of the present tumor (benign or malignant).

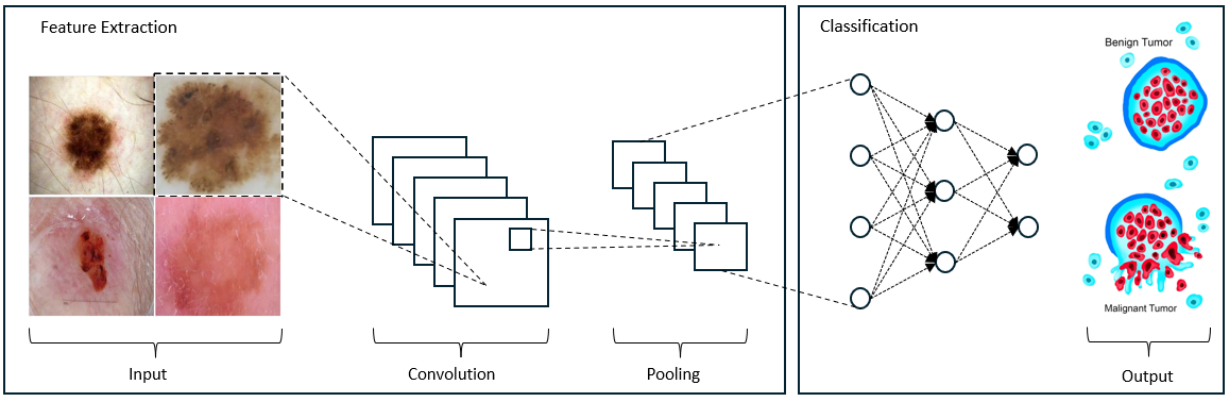


Figure 1: Basic arquitecture of a CNN

Before building the model, we performed the following preprocessing steps:

1. **Resizing images:** all images in the dataset were resized to the size of (240, 240) pixels, to facilitate processing by the neural network and ensure that all images have the same resolution.
2. **Pixel intensity normalization:** the pixel intensities of the images were normalized to the range of 0 to 1 by dividing each pixel value by 255, to prevent input values from becoming too large, which can lead to convergence problems during training.
3. **Data generator creation:** data generators were created using TensorFlow's ("*ImageDataGenerator*") class, responsible for loading images from disk and generating batches of images for training, validation, and testing of the neural network.
4. **Data splitting:** The images were divided into three distinct sets: training, validation, and testing, allowing for evaluating the model's performance on unseen data during training.

After completing the preprocessing, we built and trained the CNN model. Initially, we defined the necessary initial parameters for construction and training. This includes the format of the input images, the number of training epochs, and the calculation of the number of steps per epoch and validation.

```
input_shape = (240, 240, 3)
epochs = 30
epoch_steps = int(math.ceil(train_generator.samples/batch_size - batch_size))
validation_steps = int(math.ceil(validation_generator.samples/batch_size - batch_size))
print(f"Epoch steps -> {epoch_steps} | Validation steps -> {validation_steps}")
```

In the model construction, we employed two convolutional layers followed by pooling layers. We added a flattening layer and compiled the model, defining the optimizer, loss function, and evaluation metrics.

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 238, 238, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 119, 119, 32)	0
conv2d_3 (Conv2D)	(None, 117, 117, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 58, 58, 64)	0
flatten_1 (Flatten)	(None, 215296)	0
dense_2 (Dense)	(None, 128)	27,558,016
dense_3 (Dense)	(None, 1)	129

```
# Model Compilation
classifier.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy'])
```

After building, training, and evaluating the model, we decided to incorporate some enhancements to improve performance and stability:

- The ("*EarlyStopping*") callback was configured to halt training if the model's accuracy did not improve after 5 consecutive epochs, and the ("ModelCheckpoint") callback was used to save the best model.
- Dropout with a rate of 0.5 was added after the first dense layer to randomly deactivate certain neurons during training, aiming to reduce overfitting.

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 238, 238, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 119, 119, 32)	0
conv2d_3 (Conv2D)	(None, 117, 117, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 58, 58, 64)	0
flatten_1 (Flatten)	(None, 215296)	0
dense_2 (Dense)	(None, 128)	27,558,016
dropout (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

Evaluation measures

To evaluate our model, primarily designed to tackle a classification problem, the central measures of success have been accuracy and validation loss. Combined, we have reached various accurate conclusions about the model.

Even though they are simple measures, accuracy directly reflects the model's ability to correctly classify the data, which is the primary objective in a classification task, and validation loss has allowed us to consider overfitting and the model's ability to generalize. We decided to stick to these only since they provided us with good insight into the model, making it unnecessary to use F1 scores or other metrics.

In general, our methodology has been looking at the accuracy and noticing if there is a performance discrepancy between training and validation accuracy. Then, we analyze the loss, to make sure our model is not overfitting and deal with it.

In the end, we reserved an unseen (by the model) portion of the data, which is used to gauge the model's performance against real, unseen, data.

Regarding the datasets, we have utilized pre-existing datasets from Kaggle (link), doing little pre-processing except normalizing it between 0 and 1 and resizing it to a common size.

When considering system performance, we acknowledged computational efficiency. Memory usage and running time are critical, especially when scaling to larger datasets or deploying in resource-constrained environments. Although these aspects were not the primary focus, we have tried to write the optimized code possible at parts. In our case, computational constraints did influence our approach, preventing the implementation of more computationally demanding solutions like GridSearchCV.

In summary, our evaluation of the machine learning models considered accuracy, overfitting, generalization, and to a lesser extent, computational efficiency. While our models showed improved accuracy over time, challenges with overfitting and generalization were significant, underscoring the need for balanced model complexity and robust validation methods.

Approach

Deep Learning Model Architecture:

To tackle the classification task within this dataset, a convolutional neural network (CNN) architecture is employed. CNNs are renowned for their efficacy in image classification tasks, owing to their ability to automatically learn hierarchical features from raw pixel data. The model begins with convolutional layers, each followed by rectified linear unit (ReLU) activation functions, facilitating feature extraction. Subsequently, max-pooling layers are utilized to reduce spatial dimensions and capture essential features. To mitigate overfitting, dropout layers are incorporated, randomly deactivating a fraction of connections during training. The model architecture culminates in densely connected layers, where the extracted features are fed into a classification layer with a sigmoid activation function, yielding a binary classification output.

Challenges and Phenomena:

Building a deep learning system for melanoma classification poses several challenges. Firstly, addressing class imbalance within the dataset is crucial to prevent model bias towards the majority class. Secondly, ensuring model generalization across diverse melanoma manifestations, including variations in texture, shape, and color, requires careful consideration during model development. Moreover, optimizing computational resources for training large-scale datasets is essential to expedite model convergence and deployment. Additionally, phenomena present in the data, such as potential artifacts or noise, must be effectively captured to enhance model robustness and generalization.

Algorithm Selection and Tradeoffs:

In addressing the melanoma classification task, convolutional neural networks emerge as the algorithm of choice due to their efficacy in handling image data. However, tradeoffs between accuracy and efficiency arise, necessitating careful consideration of model complexity, training time, and computational resource utilization. Techniques like hyperparameter tuning and architecture modifications can optimize model performance while minimizing computational overhead.

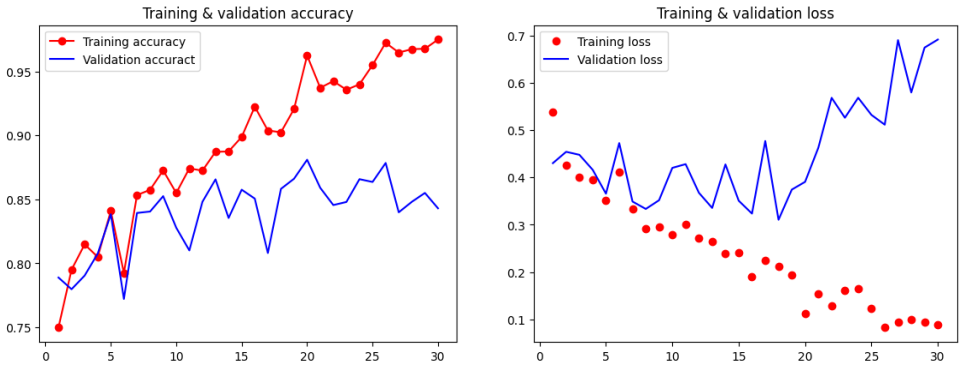
Implementation Considerations:

Implementation choices play a pivotal role in the success of the deep learning system. Preprocessing techniques, such as normalization and data augmentation, are vital to enhance model robustness and generalization. Architecture modifications, including varying depths or filter sizes, can influence model performance and computational efficiency. Additionally, hyperparameter tuning, such as adjusting learning rates or batch sizes, fine-tunes model behavior to the dataset's characteristics. By carefully considering these implementation choices, the efficacy and scalability of the deep learning system can be maximized.

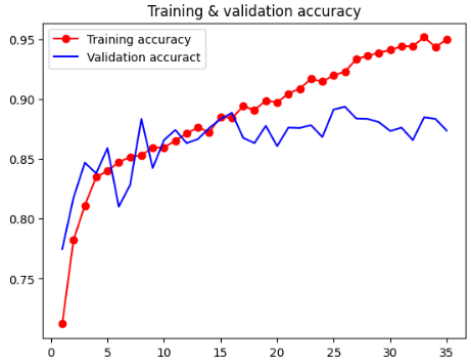
Error analysis

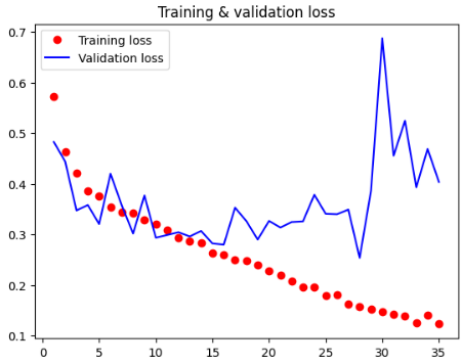
Evolution and Changes:

The initial model (Classifier (“*g4_project_final”))* showed promise learning with rising training accuracy and decreasing loss. It indicated a decent initial ability to learn from the training data. Validation accuracy also increased but not as consistently, suggesting the model was beginning to understand patterns in the data but was not yet generalizing optimally.

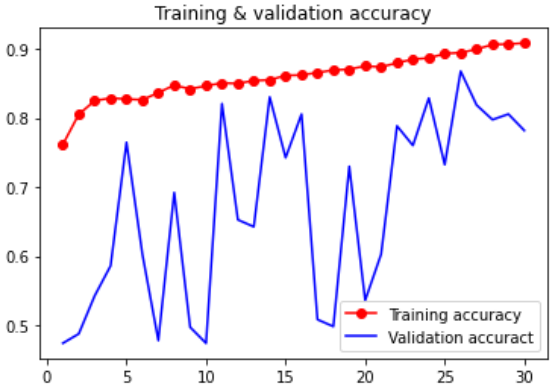


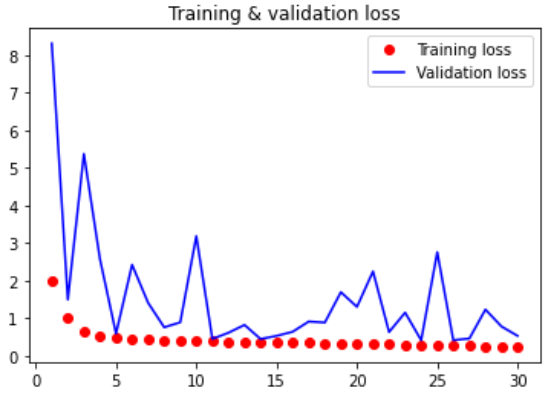
Intermediate Model (Classifier3 (“*project_attempts*”)**):** As we progressed, we observed improvements in training and validation accuracy. This indicated successful adjustments in the model architecture or training process, leading to a better fit for the data. However, validation loss began to show greater variability, which could be an early sign of overfitting in Classifier3’s case.



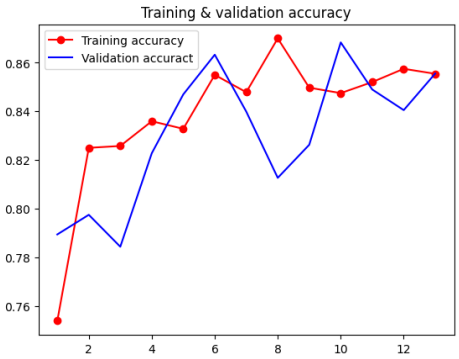


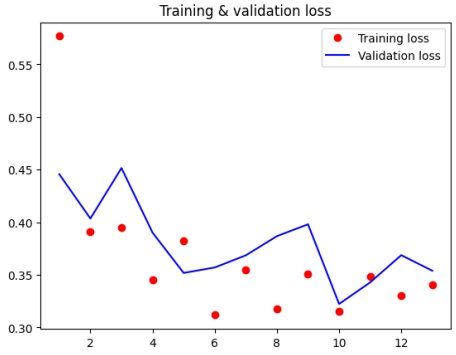
Improved Model (Classifier4 (“*project_attempts*”)**):** In the most recent models, the training accuracy became very high and stable. However, volatility in validation accuracy and loss increased significantly. This phenomenon highlighted a divergence between the model's performance on training data and validation data, a strong indicator of underfitting.





Final Model (Classifier2 (“*g4_project_final”))**:** For our final model we choose to improve our first model with the simple steps of implementing callback and dropout, the first to help with efficiency while running our models and the second one to help with our main problem throughout the assignment which was overfitting.





This final model achieved an accuracy of 87.45% when predicting on our test data that it had not seen previously as you can see in the following image.

```
Predicting test files: 100%|██████████| 2000/2000 [02:45<00:00, 12.09it/s]
Real accuracy rate: 87.45%
```

Overfitting and Generalization

Overfitting: In the later stages, the model's stability concerning training data contrasted sharply with its performance on validation data. The sharp spikes in validation loss pointed to a model that was overfitting to the nuances of the training data at the expense of its ability to generalize. This is a classic sign of overfitting, where the model learns the "noise" in the training data rather than extracting generalizable patterns.

Generalization: The oscillating validation accuracy and variable loss in the later models showed that, despite the model becoming increasingly skilled at fitting to the training data, it was losing the ability to effectively predict previously unseen data. This suggests that the learned features did not translate well outside of the training set, raising concerns about the model's generalization.

In summary, throughout the experiments, there were clear advances in fitting the model to the training data, but these advances came at the cost of reduced generalization. The most recent model appears to be overfitted, highlighting the need for a balance between deep learning of training data and maintaining robust performance on validation data. Strategies to combat overfitting, such as the introduction of regularization techniques and cross-validation, become essential to ensure the model not only learns but also generalizes well to new data.

Conclusion

In conclusion, with the development of this deep learning-based skin cancer classification system, we endeavor to contribute to the early detection and diagnosis of skin cancer, thereby enhancing patient outcomes and reducing healthcare burdens. By harnessing the power of deep learning and image analysis, the project seeks to empower healthcare professionals with advanced tools for expedited and accurate skin cancer screening and diagnosis.

Through the deployment of our system, we aspire to contribute to the early detection and diagnosis of skin cancer, thereby mitigating its impact on individuals and healthcare systems worldwide. By streamlining the diagnostic workflow and providing timely insights into skin lesion classification, our system has the potential to expedite treatment decisions, improve patient outcomes, and reduce the burden on healthcare resources.

Overall, this project sets up an image classification task, preprocesses the data, builds a CNN model, trains it, and evaluates its performance. The improvements section demonstrates the iterative nature of model development by enhancing the initial model with additional features to potentially improve its performance.