DEEP LEARNING

ALEXANDRA PINTO - 20211599
FRANCISCO FARINHA - 20211550
ILONA NACU - 20211602
JOÃO BARRADAS - 20211590
RAFAEL PROENÇA - 20211681

Abstract

This project focuses on developing a robust skin lesion classification model using a dataset from the HAM10000 competition. Leveraging a hybrid approach that merges tabular and image data, we aimed to create an ensemble model capable of accurately classifying various skin lesion types like melanoma, basal cell carcinoma, and nevi. Our approach involved building and refining an ensemble model, resulting in improved classification accuracy, particularly seen in the weighted average F1 score. Challenges arose in predicting the 'vasc' class, potentially linked to data quality, and managing overfitting while manipulating class weights. If we had more time we would explore the abcde of melanoma creating more features, transforming tabular data into images, and refining validation strategies for a more robust model. This project underscores the effectiveness of merging diverse data modalities for enhanced skin lesion classification.

1. Introduction

This project's objective revolves around developing a robust deep-learning model for a skin lesion classification task. Leveraging a dataset sourced from the HAM10000 competition, encompassing images and associated metadata, we aim to create a model capable of accurately classifying various skin lesion types, denoted by categories like melanoma, basal cell carcinoma, and nevi.

The dataset comprises images representing distinct skin lesion types, with corresponding metadata indicating essential information such as age, gender, and the type of lesion labeled under the column "dx." Our target is to create a model that performs effectively on unseen images. The available dataset includes separate folders for training and validating the model and a distinct testing set exclusively reserved for evaluating the model's performance on unseen data.

A significant challenge we encountered involved defining a methodology to effectively model both tabular data (MetaData.csv) and the images. In tackling this, we opted for a Hybrid Approach utilizing the Functional API, a concept mentioned in PowerPoint 5 in Theoretical Classes. This strategy involved creating two distinct models separately and subsequently joining them. This approach enabled us to develop and refine models using functions lectured in class.

To achieve our goal, we created four notebooks:

1. Pre-processing - Focused on data preparation and initial transformations.
2. Model CNN - Training and developing a model designed for handling image data.
3. Model NN - Training and developing a model tailored for tabular data.
4. Concatenate - Dedicated to merging and combining both models from steps 2 and 3.

Also, we decided to create a Python file named “utils.py” that contains the functions used in the 4 notebooks.

In order to aid us in the development of this project we looked for and studied some interesting articles on topics similar to our project. This would help us formulate better insights and give us ideas to improve the quality of our project.

2. Experimental Setup

In this section, we list the key Python libraries employed in our project:

- *Random*: For generating random values.
- *Pandas (as pd)*: For data manipulation and analysis.
- *NumPy (as np)*: For numerical computations and array handling.
- *OS*: For operating system-related functionalities.
- *sklearn.preprocessing.LabelEncode*r: For encoding categorical variables.
- *glob*: For file matching according to specified patterns within directories.

- *tqdm*: For creating progress bars to monitor iterative processes.
- *PIL (from PIL import Image)*: For image processing and manipulation.
- *sklearn.model_selection.train_test_split*: For splitting datasets into training and validationsubsets.
- *matplotlib.pyplot (as plt)*: For data visualization purposes.
- *keras. utils.to_categorical*: For converting class into binary class matrices,
- Functional API: provides a flexible way to build neural network models, enabling users to create architectures with shared layers, multiple inputs and outputs. We used it to create a multi-input model.
- Keras Layers: facilitate the construction of CNNs. They include functionalities for defining input layers, convolutional layers (Conv2D), pooling layers (MaxPooling2D, AveragePooling2D), flattening data (Flatten), fully connected layers (Dense), and dropout regularization (Dropout).
- TensorFlow's Functional API: empowers the creation of versatile neural network models, allowing for shared layers, multiple inputs, and outputs. In our project, we utilized this flexibility to seamlessly build a multi-input model, enhancing the adaptability of our architecture.

These libraries cover various functionalities essential for tasks like data handling, analysis, visualization, statistical testing, machine learning, and more.

3. Best approach

3.1. Pre-processing

In our metadata exploration, we delved into the distribution of images within the train/test folders and noted missing 'age' values in the metadata. To handle this absence, we employed KNN imputation for 'age' and ensured the data was converted into integers. Our visual analysis allowed us to grasp data distributions, while statistical tests like chi-squared and t-tests aided in understanding feature relevance.

For modeling, categorical features were appropriately encoded, and we introduced a new feature indicating lesion nature based on the 'dx' label. Lesions classified as benign received a value of 1, while malignant ones were assigned 0.

Regarding image data, we experimented with multiple preprocessing techniques. These methods aimed to remove hair from images, crop out black borders, extract lesion colors as a feature, and isolate relevant areas by applying a black mask. Resizing images for efficiency, we found the most effective combination: hair removal, border cropping, non-hematoma area masking, and storing images at ten percent of their original size.

The resulting images were stored as arrays, and we balanced and split metadata and image data into train/validation sets in the proportions 60%, 15%, and 25% for training, validation, and testing respectively. We established organized folders for data and models, segregating metadata and image data and ensuring categorical targets were appropriately formatted for modeling preparation.

3.2. Model implemented

In our project, we've merged tabular and image data for classification using an ensemble approach with Keras Functional API. Initially, we loaded tabular and image datasets separately. For tabular data, we encode categorical features, prepared class weights to address the imbalance, and dropped unnecessary columns. Concurrently, we loaded preprocessed image data. Then, using two separate models trained on tabular and image data, we loaded their weights into an ensemble model via the Functional API. This ensemble model combined the outputs of both models through concatenation and added dense layers for further processing before the final classification output layer. We trained this ensemble model, evaluated its performance on validation and test sets, and generated classification reports to assess its predictive accuracy.

4. Evaluation

In this section, we will first explain the metrics we used and then clear out the intermediate models we defined and the decisions we take to reach our final model.

Our main metric for evaluating model performance centered around the weighted average f1 score, primarily due to the imbalanced nature of the data. However, considering that we also want to achieve a balanced model that predicts all classes reasonably well, we also valued the individual f1 scores for each class. Relying solely on the weighted f1 score could potentially lead to a misleading assessment, so sometimes examining individual class f1 scores was crucial in ensuring a comprehensive evaluation.

The micro F1 score aggregates true positives, false positives, and false negatives across all classes to compute a single F1 score. It treats each instance equally.

The macro F1 score calculates the score for each class separately, and then the average across all classes is computed. Every class has equal weight. Whereas the weighted F1 score is a weighted average of the F1 scores for each class, where each class's score contributes proportionally to its size in the dataset. It's ideal for imbalanced datasets, as it considers class imbalance in the final score.

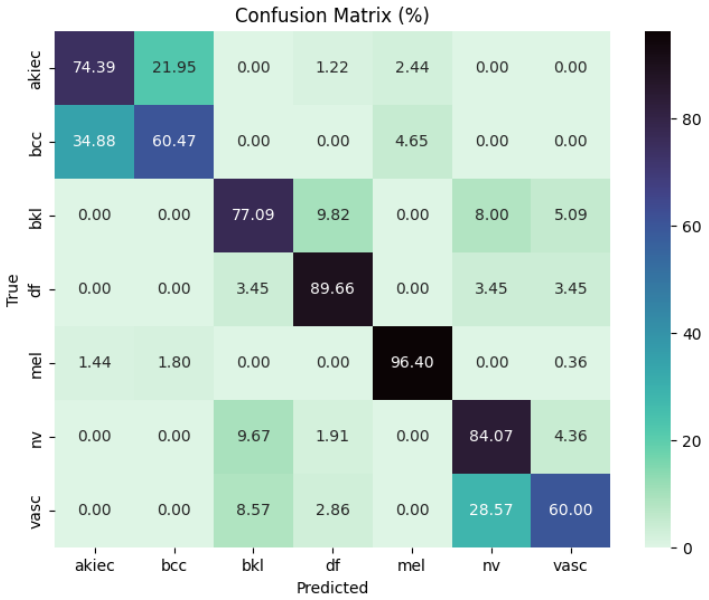
The final ensemble model combines both CNN and NN models trained on image and tabular data. These two models were also tuned to achieve the best results possible in the two models before combining them.

For CNN models we started with simpler models, then tackled overfitting problems that came up. Later tested models with more neurons and layers, exploring different structures from the usual pooling schemes. The best CNN model turned out to be the fourth one we made, using average pooling instead of maximum pooling and stacking several convolution layers in a row for its architecture.

In the NN model, we trained 5 different models to compare them and define the best one. We began by defining a simpler one first, and then we augmented the complexity and manually tuned the class weights, and in the last model we tested the capabilities of KerasTuner to determine the best parameters. The best NN model ended up being the third NN model we defined, where we manually defined class weights, which significantly impacted our unbalanced data.

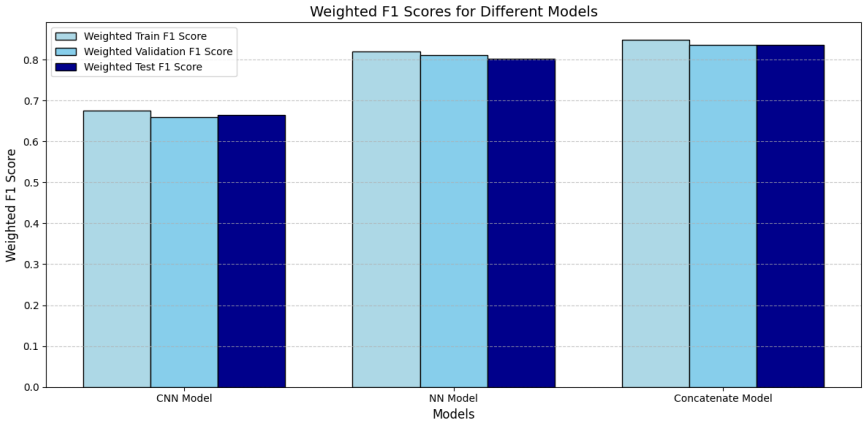
We created four ensemble models that combined the top-performing CNN and NN models. To refine their performance, we fine-tuned class weights and implemented callbacks such as reducing the learning rate on a plateau and early stopping. Ultimately, the model that performed better was the last we created, the forth.

Below we have the confusion matrix of the predictions of the best model, showcasing the percentage of the correct predictions by class.



Looking at the confusion matrix above we can extract that the model is predicting all classes reasonably well, which is what we hoped to achieve.

We compared the final ensemble model with the individual CNN and NN models that contributed to its creation. This comparison aimed to showcase the improvement achieved by combining the two models. This is illustrated in the bar plot below.



5. Error Analysis

One common problem we had in the process of our project was having difficulty in predicting ‘vasc’ class. Independently of the model, it was always hard to achieve good predictions for this class. We consider this could be caused by the lack of quality of data in this class. It came across our thoughts that could be due to the few data we had, but other classes had few data too when comparing to other classes and achieved better predictions more easily.

We also detected a phenomenon that when we manipulate class weights the models usually generate a bit of overfitting. Callbacks were useful to mitigate the impact of overfitting but some models could not avoid overfitting even when utilizing callbacks.

6. Conclusion

Our goal was to build a strong skin lesion classification model using a hybrid approach that combines tabular and image data. We achieved this by creating an ensemble model that outperformed individual CNN and NN models, showcasing improved performance in classification accuracy, particularly in the weighted average F1 score.

Challenges arose in predicting the 'vasc' class, potentially due to data quality issues, and manipulating class weights occasionally led to overfitting. If we had more time we would implement Startify k fold for a more robust model, transform tabular data into images, and explore more the abcde of melanoma creating more features. Overall, our project successfully demonstrated the effectiveness of merging tabular and image data for enhanced predictive capabilities.

7. Bibliography

1. Y. -H. Tu, I. Tashev, S. Zarar and C. -H. Lee, "A Hybrid Approach to Combining Conventional and Deep Learning Techniques for Single-Channel Speech Enhancement and Recognition," 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 2018, pp. 2531-2535, doi: 10.1109/ICASSP.2018.8461944.
2. Z. Loukil and S. AL-Majeed, "Toward Hybrid Deep Convolutional Neural Network Architectures For Medical Image Processing," 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), Sakheer, Bahrain, 2020, pp. 1-6, doi: 10.1109/3ICT51146.2020.9312027.
3. N. Briner et al., "Tabular-to-Image Transformations for the Classification of Anonymous Network Traffic Using Deep Residual Networks," in IEEE Access, vol. 11, pp. 113100-113113, 2023, doi: 10.1109/ACCESS.2023.3323927.
4. K. Foysal Haque, F. Farhan Haque, L. Gandy and A. Abdelgawad, "Automatic Detection of COVID-19 from Chest X-ray Images with Convolutional Neural Networks," 2020 International Conference on Computing, Electronics & Communications Engineering (iCCECE), Southend, UK, 2020, pp. 125-130, doi: 10.1109/iCCECE49321.2020.9231235.