MELANOMA CLASSIFICATION

Salma Asaad

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Skin cancer is the out-of-control growth of irregular skin cells causing the skin to rapidly multiply and form malignant tumors. Between 2008 and 2019 the number of new invasive Melanoma cases rapidly increased by 54%. With such a substantial escalation, it is of great importance for us to inaugurate and build on techniques to assist healthcare providers in diagnosing skin cancer at its preliminary stages.

I developed a classification model for classifying the images to Melanoma or not motivated by the performance of Convolutional Neural Networks in computer vision, I present a CNN-based model from scratch. The results show that the system performs a training accuracy of 97.5% and testing accuracy 75%. In the testing stage the model predicted 18 images correctly, as it classified them from the Nevus type (class 0), and they are, but it failed to predict 6 images from the Melanoma (class 1).

Keywords*: Image classification, deep learning, CNN, Melanoma cancer.*

1. DESIGN

Skin cancer is the out-of-control growth of irregular skin cells causing the skin to rapidly multiply and form malignant tumors. Between 2008 and 2019 the number of new invasive Melanoma cases rapidly increased by 54%. With such a substantial escalation, it is of great importance for us to inaugurate and build on techniques to assist healthcare providers in diagnosing skin cancer at its preliminary stages.

Melanoma is one of the most aggressive types of skin cancer as it rapidly spreads to various areas of the body. With the increase and fatal nature of melanoma, it is of utmost importance to establish computer-assisted diagnostic support systems to aid physicians in diagnosing skin cancer [1].

2. DATASET

The PH² dataset has been developed for research and benchmarking purposes, in order to facilitate comparative studies on both segmentation and classification algorithms of dermoscopic images. PH² is a dermoscopic image database acquired at the Dermatology Service of Hospital Pedro Hispano, Matosinhos, Portugal [2].

The dermoscopic images were obtained at the Dermatology Service of Hospital Pedro Hispano (Matosinhos, Portugal) under the same conditions through the Tuebinger Mole Analyzer system using a magnification of 20x. They are 8-bit RGB color images with a resolution of 768x560 pixels [2].

This image database contains a total of 200 dermoscopic images of melanocytic lesions, including 80 common nevi, 80 atypical nevi, and 40 melanomas. The PH² database includes medical annotation of all the images namely medical segmentation of the lesion, clinical and histological diagnosis, and the assessment of several dermoscopic criteria (colors; pigment network; dots/globules; streaks; regression areas; blue-whitish veil) [2].

The assessment of each parameter was performed by an expert dermatologist, according to the following parameters in Table 1:

Table 1:Dataset parameters.

Criterion	PH^2 Segmentation
Clinical Diagnosis	0 - Common Nevus
	1 - Atypical Nevus
	2 - Melanoma
Lesion Segmentation	Available as a binary mask (with the same size as the original image).
Color Segmentation	Available as a binary mask (with the same size as the original image) (If available).
Asymmetry	0 - Fully Simmetry
	1 – Asymmetry in One Axis
	2 - Fully Asimmetry
Pigment Network	AT - Atypical
	T - Typical
Dots/Globules	A - Absent
	AT - Atypical
	T - Typical
Streaks	A - Absent
	P - Present
Regression Areas	A - Absent
	P - Present
Blue Whitish Veil	A - Absent
	P - Present
Colors	1 - White
	2 - Red
	3 - Light-Brown
	4 - Dark-Brown
	5 - Blue-Gray
	6 - Black

3. ALGORITHMS

3.1 Data Loading and Pre-processing

3.1.1 Convert text data file to DataFrame

Load text data file to the work as a DataFrame using pandas and extract the text from the file by reading it line by line using ‘os’ library.

```
[['Name',
  'Histological Diagnosis',
  'Clinical Diagnosis',
  'Asymmetry',
  'Pigment Network',
  'Dots/Globules',
  'Streaks',
  'Regression Areas',
  'Blue-Whitish Veil',
  'Colors'],
 ['IMD003', '', '0', '0', 'T', 'A', 'A', 'A', 'A', '4'],
 ['IMD009', '', '0', '0', 'T', 'A', 'A', 'A', 'A', '3'],
 ['IMD016', '', '0', '0', 'T', 'T', 'A', 'A', 'A', '3  4'],
 ['IMD022', '', '0', '0', 'T', 'A', 'A', 'A', 'A', '3'],
 ['IMD024', '', '0', '0', 'T', 'A', 'A', 'A', 'A', '3  4'],
 ['IMD025', '', '0', '0', 'T', 'T', 'A', 'A', 'A', '3'],
 ['IMD035', '', '0', '2', 'T', 'A', 'A', 'A', 'A', '2  3'],
 ['IMD038', '', '0', '0', 'T', 'T', 'A', 'A', 'A', '4  6'],
 ['IMD042', '', '0', '0', 'T', 'T', 'A', 'A', 'A', '3  4'],
 ['IMD044', '', '0', '0', 'T', 'T', 'A', 'A', 'A', '4  5'],
 ['IMD045', '', '0', '0', 'T', 'T', 'A', 'A', 'A', '3'],
 ['IMD050', '', '0', '0', 'T', 'T', 'A', 'A', 'A', '3'],
 ['IMD092', '', '0', '0', 'T', 'T', 'A', 'A', 'A', '3  4'],
 ['IMD101', '', '0', '0', 'T', 'A', 'A', 'A', 'A', '3'],
 ['IMD103', '', '0', '0', 'T', 'T', 'A', 'A', 'A', '3  4'],
```

Figure 1: PH2 text data file.

	Name	Histological Diagnosis	Clinical Diagnosis	Asymmetry	Pigment Network	Dots/Globules	Streaks	Regression Areas	Blue-Whitish Veil	Colors
0	IMD003		0	0	T	A	A	A	A	4
1	IMD009		0	0	T	A	A	A	A	3
2	IMD016		0	0	T	T	A	A	A	3 4
3	IMD022		0	0	T	A	A	A	A	3
4	IMD024		0	0	T	A	A	A	A	3 4

Figure 2:PH2 dataset as a DataFrame.

3.1.2 Drop all (typical Nevus) records

Remove all rows that belong to the "1" Atypical Nevus and keep only "0" (common Nevus) & "2" (Melanoma). After these rows were removed, the data contained only 120 rows.

3.1.3 Load images from the directory

Listing all the image files paths and names.

```
img_name = []
path_img = []

for img in imgs :
    for img_filename in os.listdir('PH2 Dataset images/'+ img +'/' + img+'_Dermoscopic_Image' ):
        path_img.append('PH2 Dataset images/'+ img +'/' + img+'_Dermoscopic_Image'+ '/' +img_filename)
        img_name.append(img_filename)
```

Figure 3: Load image.

3.1.4 Replace column (image) with the total name of dataset images

```
In [16]:
#convert list of images to dataframe
df_img = pd.DataFrame(img_name ,columns=['Name'])
```

```
In [17]:
df_img
```

Out[17]:

	Name
0	IMD003.bmp
1	IMD009.bmp
2	IMD016.bmp
3	IMD022.bmp
4	IMD024.bmp
...	...
115	IMD424.bmp
116	IMD425.bmp
117	IMD426.bmp
118	IMD429.bmp
119	IMD435.bmp

120 rows × 1 columns

Figure 4: Rename "Name" Column.

3.1.5 Create DataFrame for the target column

In the target column (Clinical Diagnosis) I replace the ‘2’ by ‘1’ so that the target column become 1 for (Melanoma) and 0 for (common Nevus).

	Clinical Diagnosis
0	0
1	0
2	0
3	0
4	0
...	...
115	2
116	2
117	2
118	2
119	2

120 rows × 1 columns

	Clinical Diagnosis
0	0
1	0
2	0
3	0
4	0
...	...
115	1
116	1
117	1
118	1
119	1

120 rows × 1 columns

Figure 5: Before and after handling the target column.

3.2 Splitting Dataset to Train and Test sets

The total size of the dataset is 120 images, which have been tabulated in the DataFrame, containing two columns, the first one is the name of the image, and the other is the target value. In the splitting stage, the datasets will be separated by 80%:20% for training and testing sets respectively. The small one is used for testing and evaluating the model while the largest sets are used in training the model.

```
XTrain shape =  (96, 2)
yTrain shape =  (96,)
XTest sets shape =  (24, 2)
yTest sets shape =  (24,)
```

Figure 6: Datasets shapes after splitting.

3.3 Dataset Generation and Image Processing

1. **Load** the original input images from the directory.
2. **Image Processing** - Each image needs to be preprocessed before going to the classifier by resizing all the images into a high-dimensional 150 × 150 × 3 matrices based on its RGB values on each pixel, using CV2. Then store the new images in the new directory
3. **Convert** data to ImageDataGenerator from the new directory.
4. **Rescale** pixels values by 255.

Table 2: data generation parameters.

Dataset	Color mode	Class mode	Shuffling	Image size	Scaling	Batch size
Training	RGB	binary	TRUE	(150,150)	TRUE	8
Testing	RGB	binary	FALSE	(150,150)	TRUE	1

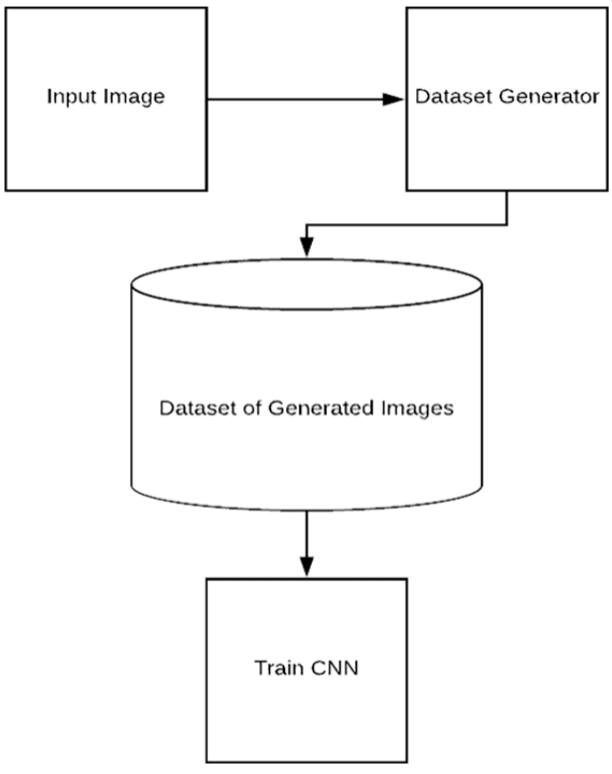

Figure 7: Data generation workflow.

```python
datagen = ImageDataGenerator(rescale=1 / 255.0)

train_generator = datagen.flow_from_dataframe(
    dataframe= X_train,
    directory='./IMages_Train_ph2/',
    x_col='Name',
    y_col="Clinical Diagnosis",
    target_size=(150, 150),
    color_mode="rgb",
    batch_size=8,
    class_mode="binary",
    subset='training',
    shuffle=True,
    seed=42
)


test_generator=datagen.flow_from_dataframe(
    dataframe= X_test,
    directory='IMages_Test_ph2/',
    x_col='Name',
    y_col=None,
    batch_size=1,
    seed=42,
    shuffle=False,
    class_mode=None,
    target_size=(150, 150 ) ,
    color_mode="rgb")
```

```
Found 84 validated image filenames belonging to 2 classes.
Found 24 validated image filenames.
```

Figure 8: Data generation Implementation.

3.4 Model Architecture

Among deep neural networks (DNN), the convolutional neural network (CNN) has demonstrated excellent results in computer vision tasks, especially in image classification. Convolutional Neural Network (CNN, or ConvNet) is a special type of multi-layer neural network inspired by the mechanism of the optical and neural systems of humans.

A CNN is a framework developed using machine learning concepts. CNN's are able to learn and train from data on their own without the need for human intervention. There is only some pre-processing needed when using CNNs. They develop and adapt their own image filters, which have to be carefully coded for most algorithms and models. CNN frameworks have a set of layers that perform particular functions to enable the CNN to perform these functions [3].

Table 3: Hyper-parameters values.

Hyperparameter	Optimizer	Learning rate	Cost function	Hidden AF	Output AF	n_epochs
Value	Adam	0.001	Binary Cross entropy	ReLU	Sigmoid	20

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 150, 150, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 75, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 37, 37, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 18, 18, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 18, 18, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 20736)             0
_________________________________________________________________
dense (Dense)                (None, 128)               2654336
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
_________________________________________________________________
flatten_1 (Flatten)          (None, 1)                 0
=================================================================
Total params: 2,683,105
Trainable params: 2,683,105
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 9: Model Architecture.

3.5 Model Training

Using model.fit_generator to pass data to the model. I will pass train and test data to fit_generator. After 20 epochs of the training process, we get a training accuracy of 97.79% as shown in Figures 11 and 12.

```
12/12 [==============================] - 1s 92ms/step - loss: 0.1978 - accuracy: 0.9167
0.0000e+00
Epoch 13/20
12/12 [==============================] - 1s 91ms/step - loss: 0.1574 - accuracy: 0.9375
0.0000e+00
Epoch 14/20
12/12 [==============================] - 1s 93ms/step - loss: 0.1816 - accuracy: 0.9062
0.0000e+00
Epoch 15/20
12/12 [==============================] - 1s 94ms/step - loss: 0.1173 - accuracy: 0.9583
0.0000e+00
Epoch 16/20
12/12 [==============================] - 1s 95ms/step - loss: 0.1338 - accuracy: 0.9375
0.0000e+00
Epoch 17/20
12/12 [==============================] - 1s 98ms/step - loss: 0.1880 - accuracy: 0.9062
0.0000e+00
Epoch 18/20
12/12 [==============================] - 1s 103ms/step - loss: 0.1639 - accuracy: 0.9271
0.0000e+00
Epoch 19/20
12/12 [==============================] - 1s 102ms/step - loss: 0.1332 - accuracy: 0.9479
0.0000e+00
Epoch 20/20
12/12 [==============================] - 1s 101ms/step - loss: 0.0646 - accuracy: 0.9792
0.0000e+00
```

Figure 10: Training phase results.

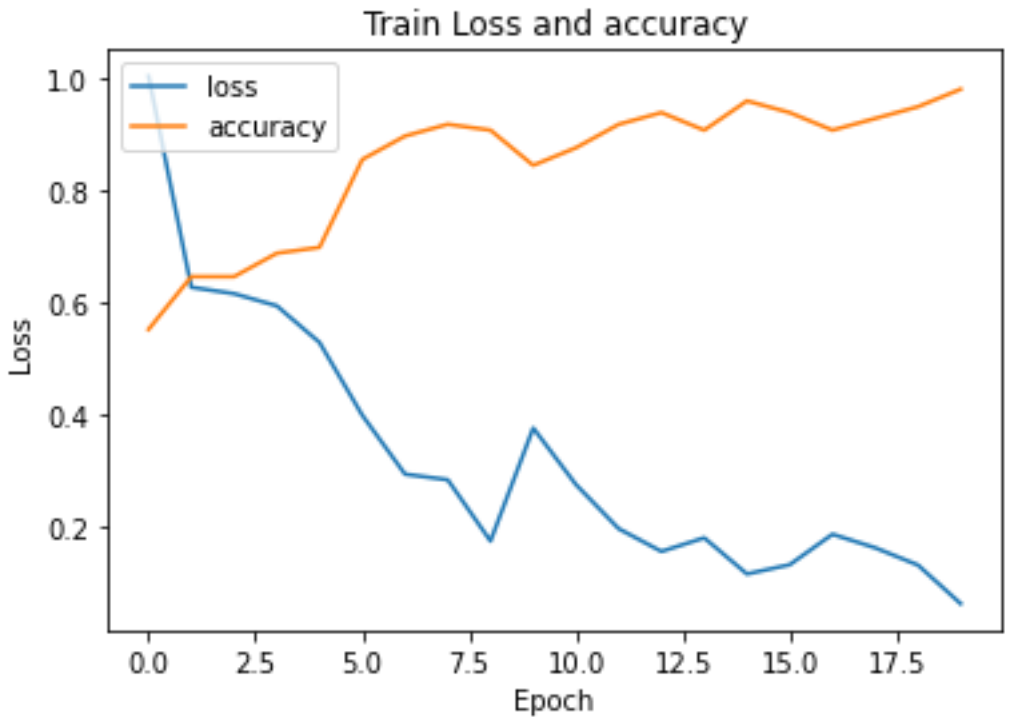

Figure 11: Training loss and accuracy graph.

3.6 Model Testing

Using the predict_generator function on 24 testing images we got these results.

- Y Actual = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0]
- Y prediction = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

3.7 Model Evaluation

3.7.1 Accuracy score.

In binary classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in the dataset.

$$Accuracy = \frac{true\ positives + true\ negatives}{total\ eamples}$$

3.7.2 Precision score.

Precision is the ability of the classifier not to label as positive a sample that is negative. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The best value is 1 and the worst value is 0.

$$Precision = \frac{true\ positives}{true\ negatives + false\ positive}$$

3.7.3 Recall score.

Recall is the ability of the classifier to find all the positive samples. The recall is intuitively the ability of the classifier to find all the positive samples. The best value is 1 and the worst value is 0.

$$Recall = \frac{true\ positives}{true\ positive + false\ negatives}$$

3.7.4 F1 score

Also known as balanced F-score or F-measure. It can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$F1 = 2\,\frac{precision\ x\ recall}{(precision + recall)}$$

3.7.5 Confusion Matrix

The model was able to predict 18 images correctly, as it classified them from the Nevus type (class 0), and they are, but it failed to predict 6 images from the Melanoma (class 1)

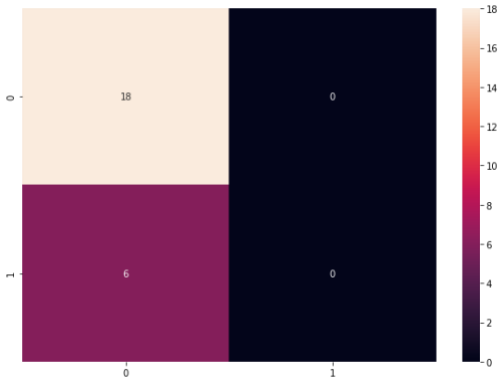

Figure 12: confusion metrics.

Table 4: The Model Results

Metric	**Accuracy**	**Precision**	**Recall**	**F1 score**	**MCC**
Value	075%	75%	75%	75%	0

4. TOOLS

- Python 3.7.3
- TensorFlow 2.4.0 and Keras
- Sklearn
- NumPy and Pandas
- Seaborn and Matplotlib
- OpenCV

5. REFERENCES

[1] Pillay V., Hirasen D., Viriri S., Gwetu M. (2020) Melanoma Skin Cancer Classification Using Transfer Learning. In: Hernes M., Wojtkiewicz K., Szczerbicki E. (eds) Advances in Computational Collective Intelligence. ICCCI 2020. Communications in Computer and Information Science, vol 1287. Springer, Cham. https://doi.org/10.1007/978-3-030-63119-2_24

[2] *ADDI - Automatic computer-based Diagnosis system for Dermoscopy Images*. (n.d.). ADDI. Retrieved November 29, 2021, from https://www.fc.up.pt/addi/ph2%20database.html

[3] Boesch, G. (2021, October 19). *A Complete Guide to Image Classification in 2021*. Viso.Ai. Retrieved November 30, 2021, from https://viso.ai/computer-vision/image-classification/

[4] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed., Vol. 2) [E-book]. O'Reilly Media.