Skin Cancer Diagnosis with Neural Networks

by

Barbara Kéri: AR5KHR
Benjámin Csontó: JTB4Y1
Sámuel Csányi: I7ULKV

Budapest University of Technology and Economics

December 2024

Chapter 1

Introduction

1.1 Skin cancer

Skin cancer is the most common type of cancer [1], but what exactly is it? Skin cancer is the out-of-control growth of abnormal cells in the epidermis, the outermost skin layer, caused by unrepaired DNA damage that triggers mutations. These mutations lead the skin cells to multiply rapidly and form malignant tumors. The main types of skin cancer are basal cell carcinoma (BCC), squamous cell carcinoma (SCC), melanoma and Merkel cell carcinoma (MCC) [2]. Melanoma is much less common than the other types but much more likely to invade nearby tissue and spread to other parts of the body. Most deaths from skin cancer are caused by melanoma[3].

The two main causes of skin cancer are the sun's harmful ultraviolet (UV) rays and using UV tanning beds. The good news is that if skin cancer is caught early, your dermatologist can treat it with little or no scarring and high odds of eliminating it entirely. Often, the doctor may even detect the growth at a precancerous stage, before it has become a full-blown skin cancer or penetrated below the surface of the skin.

Skin cancers can look quite different from one person to another due to skin tone, size and type of skin cancer and location on the body. In this project, we want to help you to recognize harmful tissues in the comfort of your home.

1.2 State-of-the-art

Machine learning has been applied to image processing for a long time. Convolutional neural networks are proven to be the most efficient in these tasks. However, studies present various architectures and explore the performance of simpler ML techniques.

A range of algorithms are explored by Shah et al[4], such as K Nearest Neighbours, SVMs and various neural network architectures. The results show that deep-learning approaches perform best, with CNNs and some expert models reaching an especially high accuracy.

Data augmentation is often applied. Thurnhofer-Hemsi et al[5] address this issue by randomly flipping horizontally, vertically or rotating the image. Another common problem that they also address is imbalanced data. It is often the case that one class is under or overrepresented in the dataset. Data augmentation can soften this imbalance, but additional techniques might be needed, which we will detail later.

Generally, the input for these models consists only of the images. This makes sense because if we want to automatize the diagnosis procedure, human labour and additional measurements should be minimized. Our dataset is special in this case since it also has a set of metadata attached to each image, with information on the patent and the parameters of the area in question on the image. This is described in greater detail in the next section.

1.3 The Dataset

The dataset consists of histologically confirmed skin cancer cases with single-lesion crops from 3D total body photos (TBP). There are 401059 images in total. The image quality resembles close-up smartphone photos, which are regularly submitted for telehealth purposes. The dataset consists of 401059 images and among their corresponding labels (positive or negative), various metadata about the patient and the leisure. We've narrowed down the features to the ones that should be useful in machine learning applications. They are shown in the table below.

Name	Description
age_approx	Approximate age of the patient at the time of imaging.
sex	Sex of the person.
tbp_lv_areaMM2	Area of lesion (mm^2).
tbp_lv_area_perim_ratio	Border jaggedness, the ratio between lesions perimeter and area.
tbp_lv_color_std_mean	Color irregularity, calculated as the variance of colours within the lesion's boundary.
tbp_lv_deltaLBnorm	Contrast between the lesion and its immediate surrounding skin.
tbp_lv_location	Classification of anatomical location divides arms and legs to upper and lower; torso into thirds.
tbp_lv_minorAxisMM	Smallest lesion diameter (mm).
tbp_lv_nevi_confidence	Nevus confidence score (0-100 scale) is a convolutional neural network classifier estimated probability that the lesion is a nevus.
tbp_lv_norm_border	Border irregularity (0-10 scale); the normalized average of border jaggedness and asymmetry.
tbp_lv_norm_color	Color variation (0-10 scale); the normalized average of colour asymmetry and colour irregularity.
tbp_lv_perimeterMM	Perimeter of the lesion (mm).
tbp_lv_radial_color_std_max	Color asymmetry, a measure of the asymmetry of the spatial distribution of colour within the lesion.
tbp_lv_symm_2axis	Border asymmetry; a measure of the asymmetry of the lesion's contour about an axis perpendicular to the lesion's most symmetric axis.
tbp_lv_symm_2axis_angle	Lesion border asymmetry angle.

Chapter 2

Data processing

Not all images are of the same dimensionality. The most straightforward solution is to rescale the images to match the most frequent dimensions. This solution shouldn't significantly worsen the models' performance as long as all the images are n by n-sized. The resulting images are 133x133 pixels with 3 RGB channels. The images are scaled to the [0-1] interval for more efficient training.

Metadata comes in a variety of data types. To create a pipeline that is compact and reusable we rely on sklearn's Pipeline API. We differentiate between categorical, binary and numerical data and apply relevant preprocessing methods: imputing is applied for each of them, to fill out missing values; numerical values are standardized (scaled to 0 mean and 1 standard deviation); binary values are binary encoded (0 or 1); categorical values are one-hot encoded.

2.1 Dynamic data loading

The dataset in processed form is too big to fit into memory: $133 * 133 * 3 * float32$. Even if we used a data type with smaller precision than 32 bits, it's difficult to manage. There are multiple ways to handle dynamic data loading in TensorFlow.

- **Python generators**
  With a generator function, you can define the process of loading your data. This gives the most flexibility as you are allowed to use any libraries and logic you want. At the same time, without TF functionality most of the optimization has to be done by the user and every step is implemented from scratch.

- **PyDataset**
  Coming from the tf.keras.utils package PyDataset is essentially a built-in version of a generator. The advantage is that you can harness the multiprocessing functionality from TF while still having a lot of flexibility.

- **Tensorflow Datasets**
  The Dataset from the tf.data API provides many features out of the box. It op-

erates on Dataset objects which can be created with a set of utilities. They are a very optimized way of creating a processing pipeline but are also limited to TF operations since they require graph execution mode.

Our solution uses *PyDataset* as a base class for our dataset, and let it handle the dynamic loading of the data.

2.2 Balancing the class samples

Positive samples are heavily under-represented, which needs to be balanced out. We use the following techniques to compensate:

- **Upsampling**
  Datapoints which belong to the positive samples are added to the dataset multiple times. This is indicated by the ‘upscale_factor‘ parameter when calling the ‘upscale_metata()‘ method.
- **Data augmenting**
  To make the upsampled images more unique, some image augmentation techniques are applied. In particular horizontal and vertical mirroring and cropping then rescaling the images. Either one or two methods are applied randomly.
- **Sample weights**
  For each sample, the loss function is evaluated using a corresponding weight, which is higher for the positive samples. We use to following formula: $c_d/(2*c_s)$, where c_d is the count of all samples and c_s is the count of samples for a given class of labels.

Chapter 3

Neural Network Architectures

3.1 Autoencoder

If we want to utilize the metadata as our input, we need to create some kind of embedding for our images, to which we can append the metadata. There are many ways to do this. The one we're going to explore is training an Autoencoder and using the encoder the create a representation of our inputs, to which we can append the relevant metadata.

The encoder part consists of three convolutional layers. The decoder has three transposed convolutions with corresponding filters to the encoder. Finally, the output is another convolutional layer that is cropped to fit the size of the labels (which in this case are the same as the input). The evaluation resulted in a mean squared error of $1,06 * 10^{-3}$. Figure 3.1 shows the encoded and reconstructed images.

For the actual classification, we embed the encoder into the data loader. The output will be flattened and the processed metadata concatenated to the result. With this, we can simply train a classifier of fully connected layers. The classifier contains a skip-connection in the beginning to mitigate the effects of gradient problems.

3.1.1 Evaluation

The model reached 80.18% accuracy on the test dataset. If we examine the confusion matrix (Figure 3.2), we can observe that the oversampling and the application of class weights were successful because the model outputs malignant class labels as well, instead of classifying everything as benign. However, the upsampling may have been too aggressive since it makes many mistakes on positive labels. Another way to improve the performance is to use more regulation, such as Dropout layers.

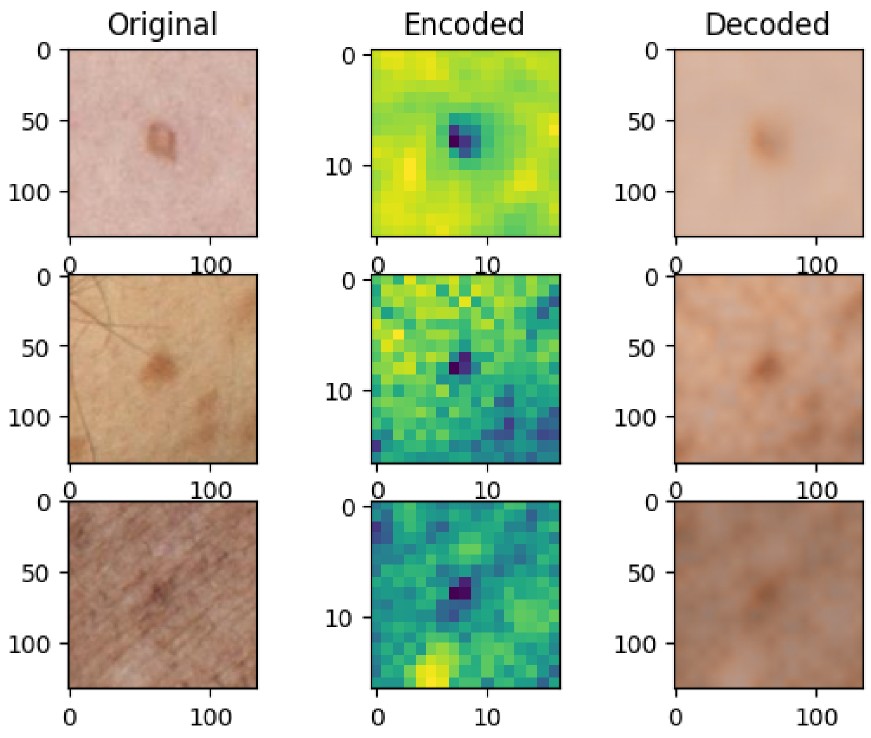

Figure 3.1: Examples of encoding and decoding images with the trained Autoencoder

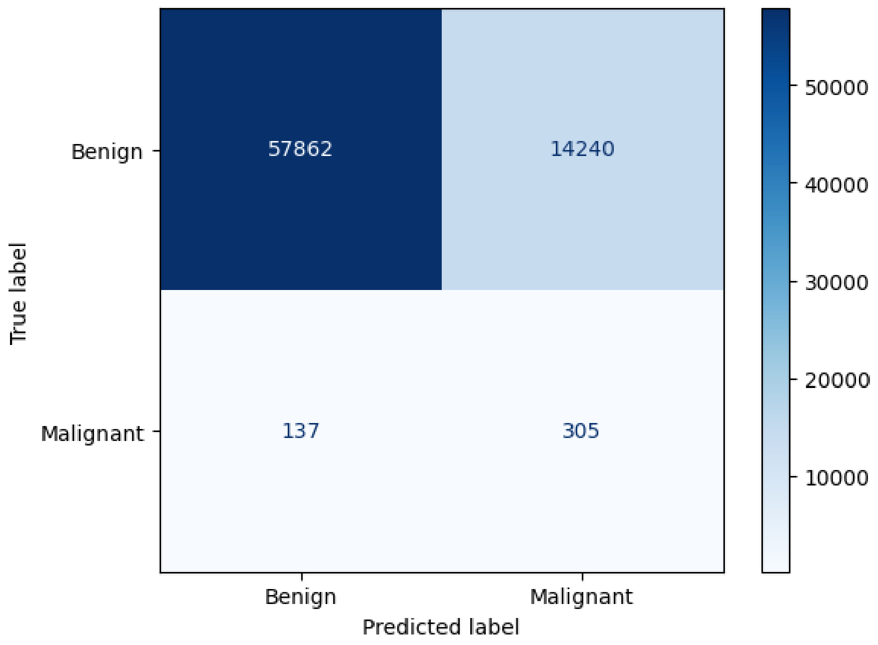

Figure 3.2: Confusion matrix about the evaluation of the test dataset

3.2 InceptionResnet

The dataset consisted of images, supplemented by extensive metadata, which we aimed to incorporate into the project. To achieve this, we used two input fields:

- Image Processing: For the images, we used the InceptionResNetV2 pre-trained network. Its high accuracy (over 80%) and relatively compact size motivated this choice.
- Metadata Processing: We implemented a simple structure comprising two standard dense layers for the metadata.

We used two separate input layers, one for the images and one for the metadata. After processing the inputs through their respective networks (InceptionResnetv2 for the images, and fully connected layers for the metadata) we concatenated the two outputs and trained an additional Dense layer on the concatenated data to get the final result. We relied on the Keras tuner for hyperparameter optimization for the Dense layers' parameters and the learning rate.

3.2.1 Evaluation

Based on the current findings, we've reached 98% accuracy, 14% precision, and 100% recall. The recall metric is the most significant because in our scenario the false negatives are more damaging than false positives (false positives can be rediagnosed, but the patients with a false negative result won't visit the doctor's office).

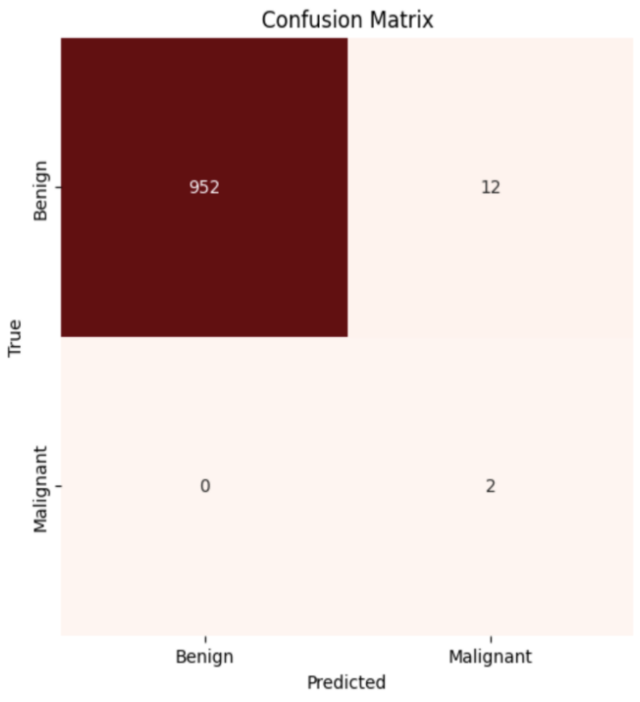

Figure 3.3: Confusion matrix of the results with the InceptionResnet

Bibliography

[1] https://www.cancer.gov/types/skin. [Online; accessed 08-Dec-2024].

[2] https://www.skincancer.org/skin-cancer-information/. [Online; accessed 08-Dec-2024].

[3] https://seer.cancer.gov/statfacts/html/melan.html. [Online; accessed 08-Dec-2024].

[4] A comprehensive study on skin cancer detection using artificial neural network (ann) and convolutional neural network (cnn). *Clinical eHealth*, 6:76–84, 2023.

[5] K. Thurnhofer-Hemsi and E. Domínguez. A convolutional neural network framework for accurate skin cancer detection. *Neural Processing Letters*, 53(5):3073–3093, Oct 2021.