

Usage of artificial neural networks on skin cancer detection

Classifying images of benign and malignant cutaneous tumors

Vasiliki Karamesiou f2821903

Evangelos Kontaratos p2821906

Erasmia Kornelatou f2821907

Christos Schismenos p2821928

ABSTRACT

We tested two neural network architectures to classify whether an image shows a benign or a malignant cutaneous tumor. Firstly, we designed a MultiLayer Perceptron neural network which consists a basic approach on image recognition/classification. The neural consisted of layers of neurons that were transferring information throughout the layers. Afterwards, we tested a Convolutional Neural Network, a more sophisticated approach on image recognition/classification. The neural network consisted of convolutional layers, which are able to extract more information from image patterns, followed by a layer of neurons that was managing the outcome of the previous layers and finally classifying the images. Finally, we compare our neural networks outcomes with an algorithm that is not dealing with neurons, the Stochastic Gradient Descent.

KEYWORDS

Skin cancer, benign, malignant, CNN, MLP, SGD, neural networks, deep learning.

1 Introduction

There are types of skin cancer that can be lethal. As an example melanoma is a type of skin cancer common in Caucasians that has large rate of mortality. Basal cell carcinoma (BCC) is the most common type of skin cancer and even if usually it is not fatal the patients having BCC use large burdens on health care services. It was estimated that 9730 deaths in 2017 are referable to melanoma [12]. As a result, the developing a machine learning algorithm that is able to classify or even predict the evolution of a tumor is of great importance.

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input [1]. This is the case when the data set consist of images. An image is not simply a sequence of bits and layers that represent the brightness of each of the basic colors (e.g. RGB model). If we have an image of a human face, then we are interested in finding face's corners and all the positions of its characteristics in order to be able to classify

whether it is a smiley, sad or angry face or to recognize if it is an authorized person that we must give access to a service. A Convolutional Neural Network is an architecture of a Deep learning class where convolutional layers extract more specific information about the examined object (which in our case is of image type).

CNNs have been developed into a powerful image classification tool. Some CNN models that have been developed are: AlexNet [9], GoogleNet [15] and Microsoft ResNet (Microsoft Research Asia, Beijing China). AlexNet have won the annual ImageNet Large Scale Visual Recognition Challenge in 2012 [9], whereas ResNet have achieved a 3.6% error rate [8]. Both achievements showing the abilities that the CNNs may have on image classification.

1.1 Previous work

In [7] Han et al. have tested ResNet-152 model on distinguishing skin cancer images into benign and malignant by parametrizing a ResNet-152 model and creating a mobile application. In [12] Rezvantlab et al. parametrized 4 neural network models DenseNet 201, ResNet 152, Inception v3, Inception ResNet v2 on detecting skin cancer achieving accuracy more than 94%. Also, many research has be done around developing neural networks on breast cancer detection [13, 14, 16]. The most preferable model that is used is CNN.

1.2 Scope

In this paper two approaches using neural networks are contained: the Multi-Layered Perceptron model and the Convolutional model. Both networks are trained on images showing two kinds of skin cancer: benign and malignant. The purpose is to create model and evaluate its ability on image classification for benign and malignant cutaneous tumors. Also a non-neuron model is developed using the Stochastic Gradient Descent method in order to be compared with the neurons-oriented approach.

2 Data

The data set [2] consists originally of 3297 images showing 1800 images of benign and 1497 images of malignant cutaneous tumors.

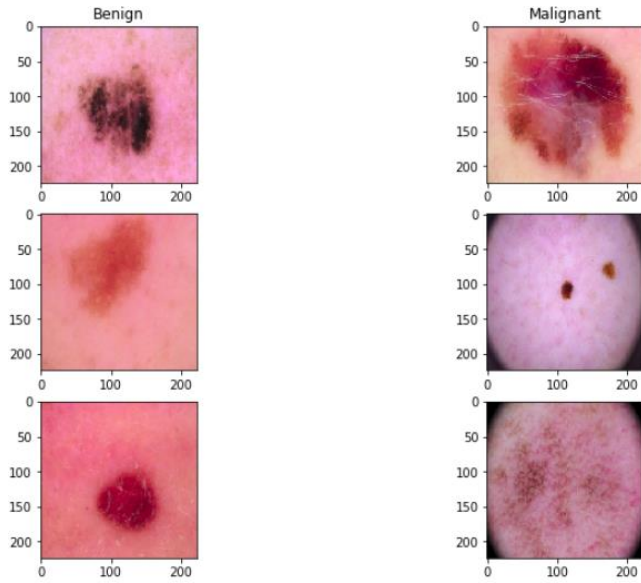


Figure 1: A sample from the data set: 3 images of benign (left column) and 3 images of malignant (right column) tumors.

2.1 Data preprocessing

For enriching the data set, the following operations are applied to the images, 4 times per image for different parameters: rotated, shifted by height/width, zoomed in/out, flipped horizontally. As an example:

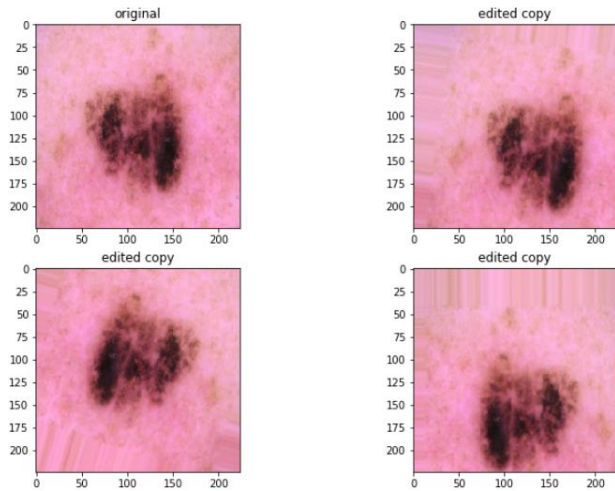


Figure 2: The original picture (upper-left) versus 3 copies of it produced by operations application.

This have been resulted an augmented data set of 13187 images (the original images were not included).

All images are of 224 width and 224 height and colored, so they consist of 3 layers (one for each of red/green/blue color).

2.2 Data structure

For the purposes of training, validating and evaluating the models that architectures selected (MLP and CNN) produce the data set was splitted to

- *train set*, which consists the 70% of the data set,
- *validation set*, which consists of 10% of the data set (used to rectify the model's parameters -such as weights- during its training) and
- *test set*, which consists of 20% of the data set (used without its ground truth labels, letting the model decide how to classify the image).

Each edited image replica was classified depending on a random variable u from uniform distribution, where if $0 \leq u \leq 0.1$ then it was considered as a validation image, if $0.1 < u \leq 0.3$, then it was considered as a test image and if $0.3 < u \leq 1$, then It was considered as a training image. The results were:

	train set	validation set	test set
images	9233	1283	2671

The split was done once and both models had the same data to be trained, fitted and evaluated (transformed properly). Due to the data set size, it was provided using flows (streams) of images [3] from the data set folder to the models (MLP, CNN and SGD). For this reason, the images were stored as following:

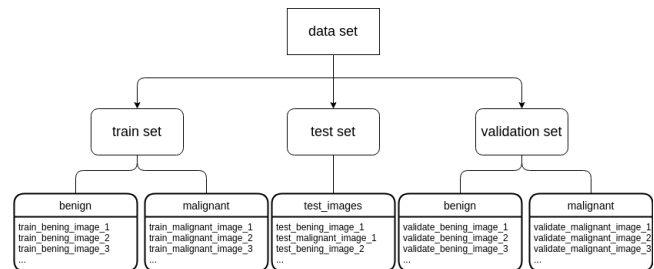


Figure 3: Data structured in order to be streamed to the models.

Furthermore, the data were normalized. Once each image is represented as a 3 dimensional array containing values $[0, 255]$ representing the brightness of each of the RGB colors, all the elements of the arrays are (true) divided by 255 in order to be normalized in the interval $[0,1]$.

3 Artificial Neural Networks

An artificial neural network simulates the way that a human brain works. It consists of:

- neurons and
- connections between the neurons.

Under this perspective, a neural network may be depicted by a directed graph having as a vertex set the set of the neurons and as the set of its edges the set of the connections in between the neurons.

Each neuron receives an amount of information and by a predefined behavior is evaluated producing an amount of information that is shared to the connected neurons.

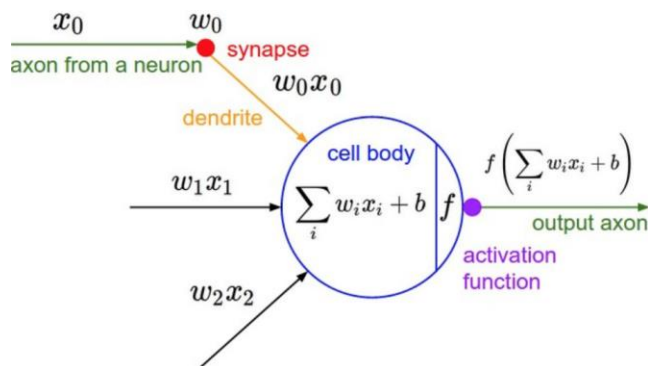


Figure 4: An artificial neuron [5]

In more details, the amount of incoming information x_0, x_1, \dots, x_n is a linear weighted combination. The weights w_0, w_1, \dots, w_n are assigned randomly at the beginning of the training process and re-adjusted per network's epoch (re-inspection of the train set). Each neuron has an activation function. Some activation functions are (the variable z below is considered to be the weighted sum mentioned above):

- **ReLU** (stands for the abbreviation of: Rectified Linear Unit) is defined as $R(z) = \max(0, z)$.
- **sigmoid** is defined as $\sigma(z) = \frac{1}{1+e^{-z}}$ which can be usage (among other usages) as a binary classification function once it is an injective function and its values lie in $[0,1]$. As a consequence, rounding to the nearest integer its outcome, turns the function to a binary classifier.
- **softmax** is defined for an input vector \mathbf{x} and vector weights \mathbf{w} , as $P(y = j \vee \mathbf{x}) = \frac{e^{\langle \mathbf{x}, \mathbf{w}_j \rangle}}{\sum_k e^{\langle \mathbf{x}, \mathbf{w}_k \rangle}}$, where $\langle \mathbf{a}, \mathbf{b} \rangle$ is the inner product of the vectors \mathbf{a} and \mathbf{b} and K is the amount of distinct classes that the sample needs to be partitioned to.

The amount $P(y = j \vee \mathbf{x})$ is the probability that the sample (which is represented by the vector \mathbf{x}) examined by the neuron to belong to class j . This function shows (normalized to $[0,1]$) in which class the sample belongs to.

If the resulting information amount for the neuron, which is produced by the composition of the activation function and the

weighted sum of the incoming information, exceeds a threshold (which varies from neuron to neuron), then this information spreads out to the neurons that are adjusted to it.

An image may be encoded in order to be network friendlier by getting transformed into a table containing numbers (an encoded information), the amount of red/green/blue (depending on the RGB layer) to the specific pixel which lies in the interval $[0, 255]$.

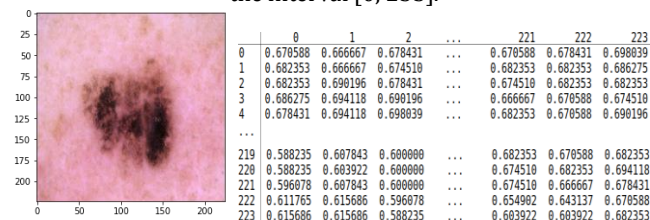


Figure 5: The human perspective (left) versus the artificial network's perspective [the green among the RGB layers was selected] (right).

Note. Hereafter, the term “neural network” will refer only to artificial neural networks.

3.1 MultiLayer Perceptron

Under the perspective of the MultiLayer Perceptron architecture, the neurons are distinguished into 3 major layers:

- The input layer
 - At this stage the data set items are transformed into a network friendly structure. Each data set item is turned into an amount that is assigned to the respective neuron.
- The hidden layer(s)
 - At this stage, there might be more than one (sub-)layers consisting of neurons of different amount and/or different behavior.
- The output layer
 - At this stage the network produces its outcome. This layer may consist neurons equal to the amount of classes that the input must be classified or neurons for evaluating the input (this is used for continuous values data).

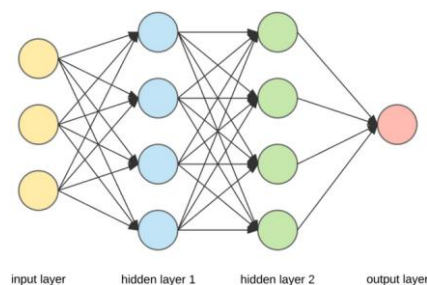


Figure 6: A sample graph representing a neural network with 2 hidden layers [4]

A neuron may share its information that produces to it neighborhood neuron(s) either to a level that is next to the neurons level (forward propagation), to a level the is prior to the neuron's level (back propagation) or to the neuron itself (feedback).

3.2 Convolutional Neural Network

Due to the fact that images are more complex bojects containing infromation (they depict other objects which have shape, corners, they are lighted or darkened), convolutional layers are used to capture these properties. Convolutional Neural Networks (CNN) are inspired by the following notion:

Definition 3.1. Given two functions $f, g: A \rightarrow B$ for $A, B \subset \mathbb{R}$ then the convolution of f and g is defined as follows:

$$f * g := \int f(\tau) \cdot g(t - \tau) d\tau$$

In a naive approach the convolution of f and g is the area of the surface the is produced in between the graph of f and the sliding of the graph of function g through the x -axis. In the same manner, each of the data entries corresponds to the function f of the Definition 3.1. A filter is defined as a matrix consisting of weights that correspond to the importance of each matrix of data entry position (at the present case a position is a bit); this corresponds to the function g of the Definition 3.1. By sliding the filter (function g) over the table that corresponds to the data entry (function f) and by applying matrix multiplication on each sliding, the neural network evaluates the specific block and deducts of the importance of each block. An example of filter application (convolution of image part and filter's) is shown below:

$$\begin{bmatrix} 6 & 5 & 4 & 3 \\ 2 & 1 & 0 & 9 \\ 8 & 7 & 6 & 5 \\ 4 & 3 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 25 & 20 \\ 15 & 14 \end{bmatrix}$$

The (1,1)-element (upper left) of the result is the product of filters application to the upper-left 3x3 block of the matrix (part of the image). We then compute:

$$6 \cdot 1 + 5 \cdot 0 + 4 \cdot 1 + 2 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 8 \cdot 1 + 7 \cdot 0 + 6 \cdot 1 = 25$$

The same procedure is applied by "sliding" the filter over the table.

After some applications of different convolutional layers, the MLP architecture is again applied to the transformed data in order to produce the final output. The overall CNN architecture looks like as follows:

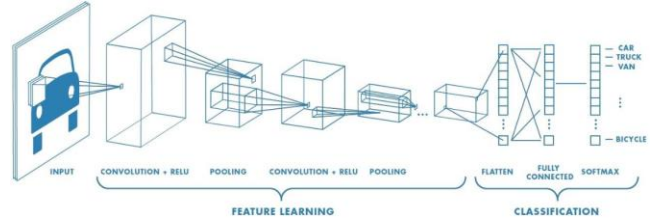


Figure 7: A view of the convolutional neural network: The images is analyzed by convolutional layers and the resulting outcome is fed to layers of neurons in order to extract the final result [5]

The application of convolutional layers is not only capturing the image properties (as mentioned in the beginning of the present subsection), but additionally makes an effective dimensional reduction of the data. Transforming an image to a vector (i.e. flattening the image) leads to vectors of high dimensionality; as an example an colored image of 224x224 is represented as a 3-dimensional array of 224x224x3 dimensions and its flattening will produce of vector of 224x224x3 = 150528 dimensions which is really hard to be manipulated.

4 Stochastic Gradient Descent

We firstly introduce the following notion:

Definition 4.1. The **partial derivative** of function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ with respect to the basis vector e_j is defined as:

$$\partial_{e_j} f := \lim_{h \rightarrow 0} \frac{f(x) - f(x + h \cdot e_j)}{h}, \quad j = 1, \dots, n$$

Generalizing the notion of the function's derivative to multivariate functions we have the:

Definition 4.2. The **gradient** of function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is:

$$\nabla f := (\partial_{e_1} f, \dots, \partial_{e_n} f), \text{ for basis } e_1, \dots, e_n \in \mathbb{R}^n$$

Given a minimization problem with objective function f , we know that a root of its gradient (derivative on 1-dimensional space) consists of a local minimum for the objective function. The Gradient Descent algorithm follows the direction of the gradient towards to the local minimum of the objective function. We may deduce that GD consists of the following algorithm:

1. do
2. batch \leftarrow data set
3. theta_grad =
 gradientEval(f, batch; theta)
4. theta = theta - alpha * theta_grad
5. until (termination criterion is met)

The parameter $0 < \alpha < 1$ consists of a hyper-parameter (i.e. it is provided by the user). The parameter theta is reduced

repeat-by-repeat in order not to bypass a local minimum (so the algorithms needs to make smaller steps -i.e. to be more cautious during its search- as long as it iterates). The characterization “stochastic” is deduced by the method’s (randomly) selection of training into batches and not over the whole data set at once.

5 Results

The initial hyper-parameters (i.e. those which are determined by the user) for the architectures that we tested were:

- 10 epochs for MLP, 10 epochs for CNN and 5 epochs for the SGD;
- initial learning rate: 10^{-5} ;
- reduce of learning rate per 3 epochs to the 1/100 of the previous epoch learning rate, but with a lower bound of 10^{-7} ;
- 2 classes (benign and malignant);
- all neurons used at the intermediate levels are having as an activation function the ReLU; the output level neurons are having the softmax activation function.

Note. The train and validation set are given shuffled as an input at the models, once they are structured as in Figure 3. The test set image are not organized per class (as train and validation set), as a result they are provided with the order that they are stored in the respective folder.

CNN is a more sophisticated architecture as about image recognition/classification, but on the other hand this demands more time in data processing. The application of the convolutional filters on the images in order to extract information is a high cost of time process. As a result in all our experiments, the training of a CNN took more time than an MLP. For the preferences that are described in the following sections there were observed the following times:

- Mean duration per period in MLP: 75.4 seconds.
 - Min duration: 73 seconds.
 - Max duration: 80 seconds.
- Mean duration per period in CNN: 260.6 seconds.
 - Min duration: 252 seconds.
- Max duration: 286 seconds.
- Mean duration per period in SGD: 99.98 seconds.
 - Min duration: 93.68 seconds.
- Max duration: 104.34 seconds.

Since we are interested in classifying objects (images), the output neurons layer at each neural network has 2 neurons (which equals the number of classes).

5.1 MultiLayer Perceptron Model

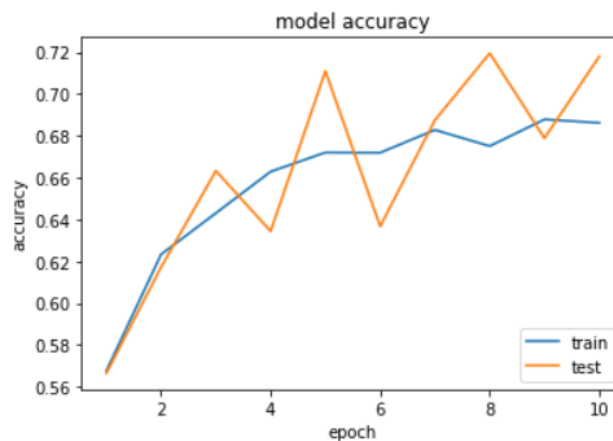
We created a MLP with:

- The input layer was consisting of 196608 neurons (which is the result of flattening the $256 \times 256 \times 3$ image -flow of images transformed the images from 224×224 to 256×256 - table representation to a vector).
- 2 dense (i.e. fully connected with the previous layer) neurons (sub-)layers in the hidden layer each of them consisting from 512 neurons.

After each layer (input and all (sub-) layers in hidden layer), the 50% of the produced information was dropped (to avoid make network overfitting, i.e. make it to imitate the train set).

		true	
		benign	malignant
predicted as	benign	1094	310
	malignant	398	869

Below it is depicted the model’s accuracy on train and on validation (referenced as “test” in diagram’s legend) sets during its training process.



By inspecting the model’s accuracy diagram during its validation process, during its validation epochs (see below) the model’s highest recorded accuracy was 71.95% at the end of its 8th epoch of validation.

Furthermore, the per class statistics were:

	precision	recall	f1-score	support
benign	0.7792	0.7332	0.7555	1492
malignant	0.6858	0.7370	0.7105	1179
accuracy	N/A	N/A	0.7349	2671
macro avg	0.7325	0.7351	0.7330	2671
weight avg	0.7380	0.7349	0.7356	2671

It is deduced (by precision column) that the 78% of (totally 1492) benign images and the 69% of (totally 1179) malignant images were classified correctly by the MLP model. What is more, the model's accuracy on test set was 74.24%.

5.2 Convolutional Neural Network

We created a CNN with:

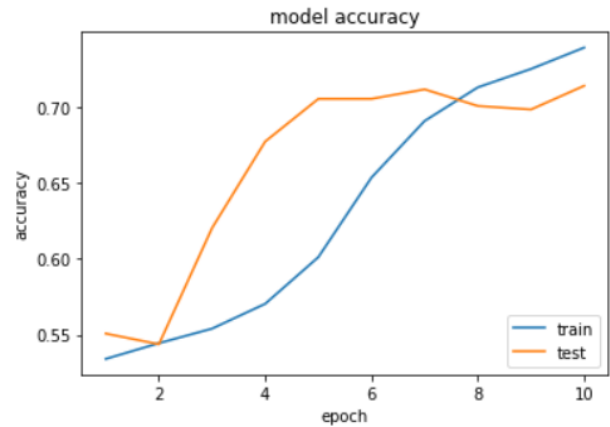
- 3 convolutional layers each of them consisting from 32 filters of 5x5 dimensionality.
- 1 dense (i.e. fully connected) layer consisting of 512 neurons.

After each convolutional layer, the 25% of the produced information was dropped (in order to avoid overfitting).

The diffusion matrix for the model founded to be:

		true	
		benign	malignant
predicted as	benign	1121	317
	malignant	371	862

The model's accuracy on validation set was 71.41% as it may be seen in the diagram of its accuracy behavior during it validation process that follows.



Furthermore, the per class statistics were:

	precision	recall	f1-score	support
benign	0.7795	0.7513	0.7651	1492
malignant	0.6991	0.7311	0.7147	1179
accuracy	N/A	N/A	0.7424	2671
macro avg	0.7393	0.7412	0.7399	2671
weight avg	0.7440	0.7424	0.7429	2671

It is deduced (by precision column) that the 78% of benign images and the 70% of malignant images were classified correctly. Moreover, the model's accuracy on test set was 74.24%.

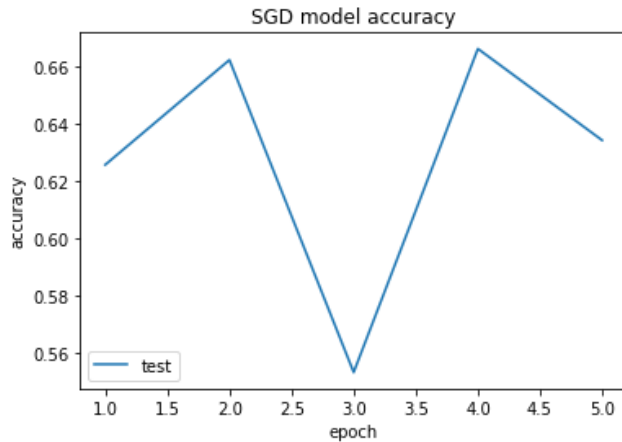
5.3 Stochastic Gradient Descent

In SGD algorithm there is not correction by the validation set as it is performed in neural networks. The validation set was used for inspecting model's accuracy during its training epochs. We created a Stochastic Gradient Descent model with reduction rate $\alpha = 10^{-5}$. The model's confusion matrix found to be:

		true	
		benign	malignant
predicted as	benign		
	malignant		

predicted as	benign	888	692
	malignant	604	487

The model's accuracy during the epochs of its training is depicted in the diagram below. Model's maximum accuracy on validation set found to be approximately 66%.



Furthermore, the per class statistics found to be:

	precision	recall	f1-score	support
benign	0.5620	0.5952	0.5781	1492
malignant	0.4464	0.4131	0.4291	1179
accuracy	N/A	N/A	0.5148	2671
macro avg	0.5042	0.5041	0.5036	2671
weight avg	0.5110	0.5148	0.5123	2671

By inspecting the precision column it may be seen that the 56.2% of benign images and the 44.64% of malignant images were classified correctly. Moreover, the model's accuracy on test set was 51.48% (the highest recorded among the models that we tried).

6 Conclusion

On our try to classify images of tumors as either benign, or malignant, we developed and configured machine learning

models. We tried two appearances of neural networks: MultiLayer Perceptron and Convolutional and a non-neural logic: Stochastic Gradient Descent. The configuration of the neural networks was a try-and-error procedure once if when we were trying to add more layers or neurons the networks were over-fitting, resulting test set to be predicted only as one class. The models overview is contained in the table that follows.

	MLP	CNN	GCD
neuron(MLP) filter size(CNN)	(512, 512)	(32, 32, 32)	N/A
# Layers	3 (3 neurons)	3 (2conv+1neur)	N/A
# Epochs	10	10	5
Mean Epoch duration (s)	75.4	260.6	99.98
Accuracy (%)	49.83	49.51	51.48

(bold letters indicate the best results in category -line)

It may be observed that:

1. Convolutional NN needed a lot of time to be trained. This was due to the convolutional layers that were applying filters to the image. On the other hand, the Perceptron model took the least mean time per epoch to be trained.
2. All our models achieved accuracy around 50%. This means that either the configurations combination (filter size, epochs, ...), or the data set was poor.

As a concluding remark our neural network models were tending to overfit. This can be deduced by inspecting the table below.

%	MLP	CNN	SGD
train-valid accuracy	74.06	76.64	~66
test accuracy	49.83	49.51	51.48

NNs were imitating the train set (i.e. high accuracy on train-validation set), making them insufficient to predict (i.e. low

accuracy on test set). On the other hand, the SGD algorithm, did not overfit, resulting a better accuracy than our NNs.

REFERENCES

- [1] Deng, L.; Yu, D., *Deep Learning: Methods and Applications* (PDF). Foundations and Trends in Signal Processing. 7 (3–4): 1–199 (2014). doi:10.1561/20000000039.
- [2] <https://www.kaggle.com/fanconic/skin-cancer-malignant-vs-benign>.
- [3] <https://keras.io/api/preprocessing/image/>
- [4] <https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8>.
- [5] https://web.stanford.edu/class/cs231a/lectures/intro_cnn.pdf
- [6] <https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb>.
- [7] Han S. S., Kim M. S., Lim W., Park G. H., Park I., Chang S. E., Classification of the Clinical Images for Benign and Malignant Cutaneous Tumors Using a Deep Learning Algorithm, doi:10.1016/j.jid.2018.01.028.
- [8] He K, Zhang X, Ren S, Sun J., *Delving deep into rectifiers: surpassing human-level performance on ImageNet classification*, <https://arxiv.org/abs/1502.01852>; 2015.
- [9] Krizhevsky A, Sutskever I, Hinton GE. *Imagenet classification with deep convolutional neural networks*. Adv Neural Inf Process Syst 2012;25:1097–105.
- [10] Lomas A, Leonardi-Bee J, Bath-Hextall F., *A systematic review of worldwide incidence of non melanoma skin cancer*. Br J Dermatol 2012;166:1069–80.
- [11] Ragab D., Sharkas M., Marshall M., Ren J., *Breast cancer detection using deep convolutional neural networks and support vector machines*, US National Library of Medicine, National Institutes of Health.
- [12] Rezvantlab A., Safigholi H., Karimijeshni S., Dermatologist Level Dermoscopy Skin Cancer Classification Using Different Deep Learning Convolutional Neural Networks Algorithms.
- [13] Siegel RL, Miller KD, Fedewa SA, Ahnen DJ, Meester RG, Barzi A, et al. Colorectal cancer statistics, 2017. CA Cancer J Clin 2017;67:177–93.
- [14] Sumaiya D., Maha M. K., Saiful I., Cancer diagnosis in histopathological image: CNN based approach, <https://doi.org/10.1016/j.imu.2019.100231>.
- [15] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A., *Going deeper with Convolutions*, available at: <https://static.googleusercontent.com/media/research.google.com/en/pubs/archive/43022.pdf>.
- [16] Yurttakal A. Erbay H., Ikizceli, T., Karacavus S., *Detection of breast cancer via deep convolutional networks using MRI images*, Springer Link.