# 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. First and foremost, we designed a MultiLayer Perceptron (MLP) neural network, which consists of a basic approach on image recognition/classification. The neural was composed of layers of neurons that were transferring information throughout the layers. Afterwards, we tested a Convolutional Neural Network (CNN), 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 were managing the outcome of the previous layers and finally classifying the images. Last but not least, we compare our neural networks outcomes with two algorithmic implementations that do not deal with neurons, the Stochastic Gradient Descent and the Support Vector Machines.

## KEYWORDS

Skin cancer, benign, malignant, CNN, MLP, SGD, SVM 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 of a machine learning algorithm that is able to classify or even predict the evolution of a tumor, is of great importance.

In order to diagnose skin cancer speedily at the earliest stage, there has been extensive research solutions by developing computer image analysis algorithms. What is more, 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 consists of images. An image is not a simply 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 show 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] Rezvantalab 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%. Nevertheless, many researchers have developed neural networks on breast cancer detection [13,14, 16]. The most preferable model is CNN.

### 1.2 Mission

This project's aim is to offer a faster, safer, more accurate and overall better option when diagnosing skin cancer. We live in a country that the sun shines throughout the year, and millions of people visit Greece to enjoy its warmth and get as tan as possible. However, prolonged sun exposure can lead to huge health risks, one of which is skin cancer. Therefore, our mission was to create a tool to aid doctors and dermatologists in our country and worldwide in forming a proper and early conclusion on the properties of the skin cancer marks and

determine whether they are concerning on not, in other words: benign or malignant. That would ultimately save thousands of lives each year and improve the overall functionality, efficiency and cost of many health care systems in that regard around the world.

In this project, two approaches using neural networks are contained: the Multi-Layered Perceptron model and the Convolutional model. Both networks were trained on images showing two kinds of skin cancer: benign and malignant. The purpose is to create models and evaluate its ability on image classification for benign and malignant cutaneous tumors. In addition, two non-neuron models were developed using the Stochastic Gradient Descent method and the Support Vector Machine 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.
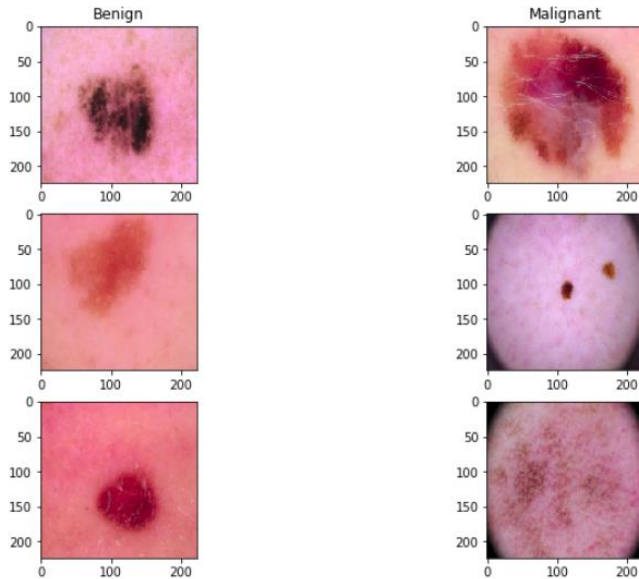


**Figure 1:** A sample from the data set: 3 images of benign (left column) and 3 images of malignant (right column) tumors.

## 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. This have been resulted in an augmented data set of 13188 images (the original images were not included). For instance:
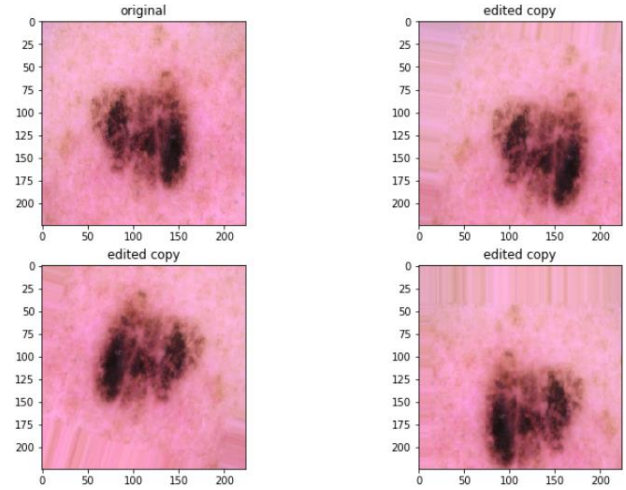


**Figure 2:** The original picture (upper-left) versus 3 copies of it produced by operations application.

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 selected models architectures (MLP and CNN) that 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 based on a random variable u from uniform distribution, where if $0 \leqslant u \leqslant 0.1$, then it was considered as a validation image, if $0.1 < u \leqslant 0.3$, then it was considered as a test image and if $0.3 < u \leqslant 1$, 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 both models had the same data to be trained, fitted and evaluated (transformed properly). The images were stored as following:
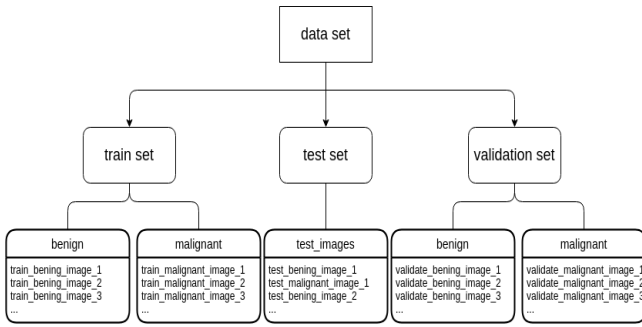


**Figure 3:** Data structured in order to be streamed to the models.

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    Methodology

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 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.
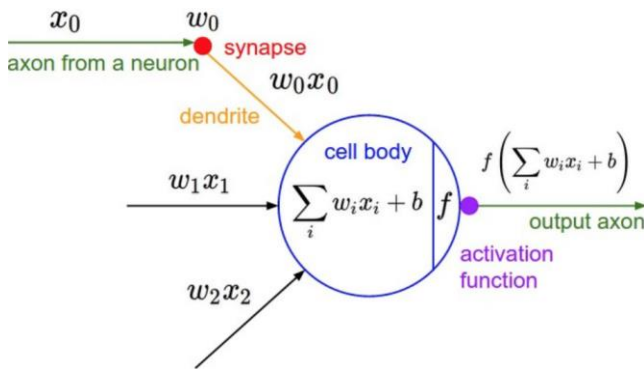
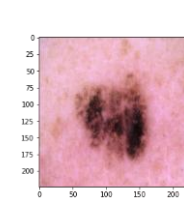

**Figure 4:** An artificial neuron [5]

In more details, the amount of incoming information $x_0, x_1, ..., x_n$ is a linear weighted combination. The weights $w_0, w_1, ..., w_n$ are assigned randomly at the beginning of the training process and re-adjusted per network's epoch (re-inspection of the train set). The re-adjustment is a procedure called 'back-propagation'. During this procedure, we compute the (partial) derivatives of the activation functions that are used on each synapse. We then, run backwards the network giving as an input to the derivatives the amount of information that the specific neuron have computed during the current epoch. Thus, running the network backwards we re-adjust this way the weights on each of the neurons. 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 the outcome to the nearest integer, turns the function to a binary classifier.
- **softmax** is defined for an input vector **x** and vector weights **w**, as $P(y = j \vee x) = \frac{e^{\langle x, w_j \rangle}}{\sum e^{\langle x, w_k \rangle}}$, where $\langle a, b \rangle$ is the inner product of the vectors **a** and **b** and K is the amount of distinct classes that the sample needs to be partitioned to.

The amount $P(y = j \vee x)$ is the probability that the sample (which is represented by the vector **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 friendly by being 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 $[0, 1]$.

**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).
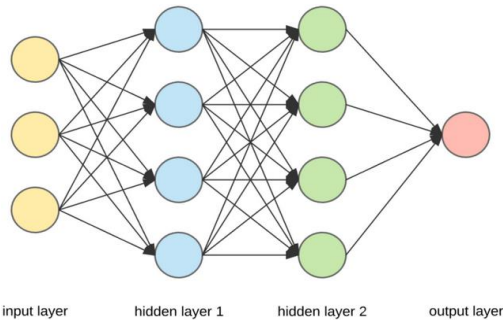


**Figure 6:** A sample graph representing a neural network with 2 hidden layers [4]

A neuron may share its information that produces to its neighborhood neuron(s) either to a level that is next to the neurons level (forward propagation), to a level that 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 objects containing information (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 $f, g: A \to B$ 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 that is produced 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 deduces 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:
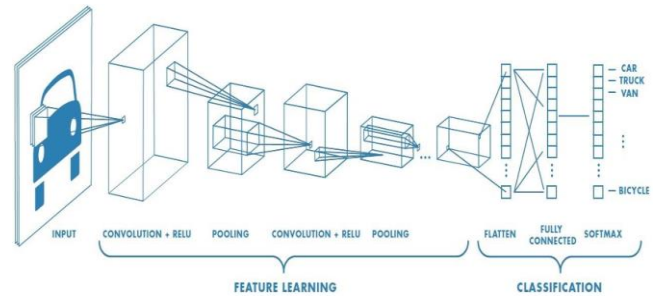


**Figure 7:** A view of the convolutional neural network: The images is analyzed by convolutional layers and the resulting

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 also 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 $f\colon \mathbb{R}^n \to \mathbb{R}$ with respect to the basis vector $\mathbf{e}_j$ is defined as:

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

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

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

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

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 ← 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 concluded by the method's (randomly) selection of training into batches and not over the whole data set at once.

## 5    Support Vector Machines

The objective of a Support Vector Machine (SVM) is to define a hyperplane so as to classify a sample with n features. By definition, a hyperplane's dimension is one less than its ambient space's dimensions. In other words, if our data samples consist of 2 features, then the SVM will design a hyperplane which will be a line; if they consist of 3 features, then the SVM will design a hyperplane which will be a 2d area; etc. Once we have classified our data, There can be infinite number of hyperplanes which resulting the same (desired) classification. The SVM algorithm choose the one hyperplane that makes the classes to be the most far from each other (with respect to a (Euclidean) metric).
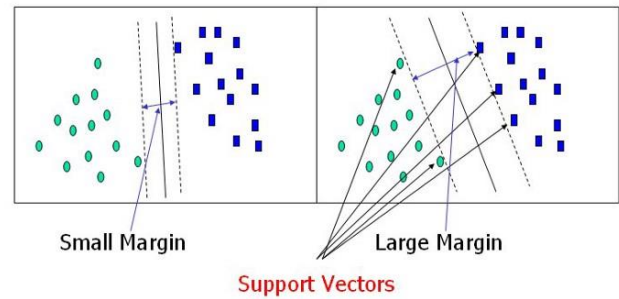


**Figure 8:** An example of hyperplane that creates a classification into two classes for the data points [17]. The SVM algorithm will prefer the one hyperplane that creates the biggest margin between separate classes.

The points of each class that are closest to the hyperplane are called **supporting vectors** (which result the algorithm's name).

The case of 2 classes is clear. When we are dealing with more than 2 classes, then there can be two approaches that we can use:

- one versus one: This approach creates one classifier for each pair of classes. The majority of the votes of the classifiers decide in which class does each data set sample belongs to.
- one versus rest: This approach creates one classifier per class which has coefficients for each of the rest classes. The class with the classifier that achieves the highest score among all the classifiers is the class in which the data set element will be classified.

We have chosen the (default) preference one versus one.

## 6    Results

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

- 18 epochs for MLP, 18 epochs for CNN, 5 epochs for the SGD and 5 epochs for SVM;
- 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 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 far as image recognition/classification is concerned. 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. Consequently, 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: 217.72 seconds.
  - ◦ Min duration: 153 seconds.
  - ◦ Max duration: 307 seconds.
- Mean duration per period in CNN: 862 seconds.
  - ◦ Min duration: 500 seconds.
  - ◦ Max duration: 1192 seconds.
- Mean duration per period in SGD: 243 seconds.
  - ◦ Min duration: 231 seconds.
  - ◦ Max duration: 255 seconds.
- Mean duration per period in SVM: 609 seconds.
  - ◦ Min duration: 222 seconds.
  - ◦ Max duration: 964 seconds.

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

### 6.1    MultiLayer Perceptron Model

We created a MLP with:

- The input layer was consisting of 196608 neurons (which is the result of flattening the 256x256x3 image

-flow of images transformed the images from 224x224 to 256x256- 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 (trying to avoid network overfitting, i.e. not to imitate the train set).

|               |            | true       |            |
|---------------|------------|------------|------------|
|               |            | benign     | malignant  |
| predicted as  | benign     | 987        | 157        |
|               | malignant  | 505        | 1022       |

Below it is depicted the model's accuracy on train and on validation sets during its training process.
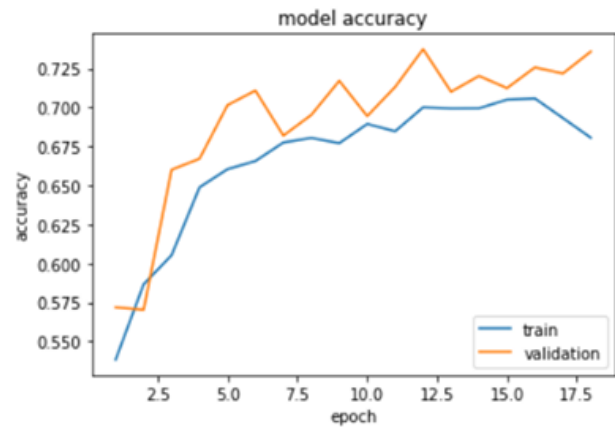


By inspecting the model's accuracy diagram during its validation process, during its validation epochs (see below), the model's highest recorded accuracy was 74 % at the end of its 12th epoch of validation.

Furthermore, the per class statistics were:

|  | precision | recall | f1-score | support |
|---|---|---|---|---|
| benign | 0.8627 | 0.6615 | 0.7488 | 1492 |
| malignant | 0.6692 | 0.8668 | 0.7553 | 1179 |
| accuracy | N/A | N/A | 0.7521 | 2671 |
| macro avg | 0.7660 | 0.7641 | 0.7521 | 2671 |
| weight avg | 0.7773 | 0.7521 | 0.7517 | 2671 |

It is deduced (by precision column) that the 86% of (totally 1492) benign images and the 67% of (totally 1179) malignant images were classified correctly by the MLP model. What is more, the model's accuracy on test set was 75%.

## 6.2 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 | 987 | 66 |
|  | malignant | 505 | 1113 |

By inspecting the model's accuracy diagram during its validation process, during its validation epochs (see below), the model's highest recorded accuracy was 78 % at the end of its 21th and 25th epoch of validation.
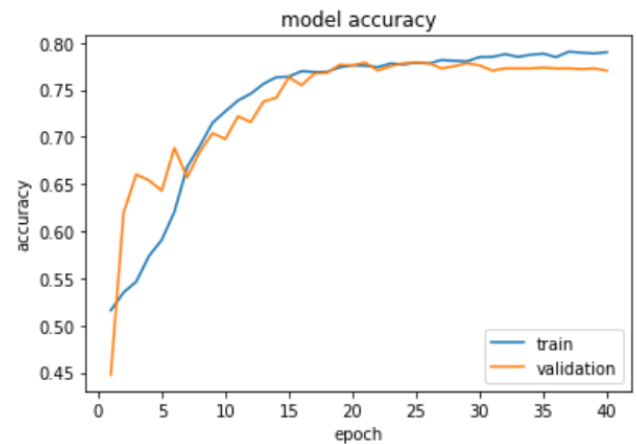


Furthermore, the per class statistics were:

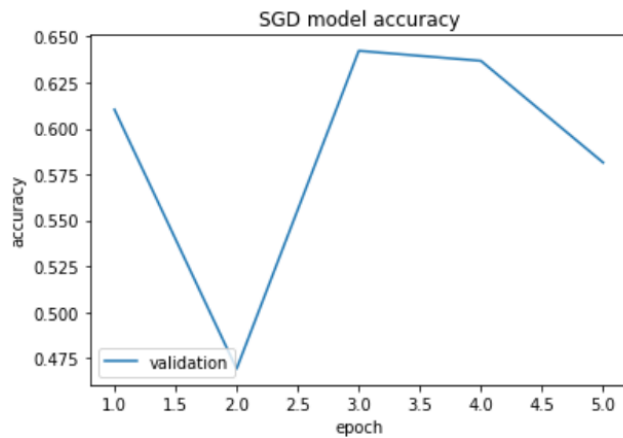|  | precision | recall | f1-score | support |
|---|---|---|---|---|
| benign | 0.9373 | 0.6615 | 0.7756 | 1492 |
| malignant | 0.6878 | 0.9440 | 0.7958 | 1179 |
| accuracy | N/A | N/A | 0.7862 | 2671 |
| macro avg | 0.8126 | 0.8027 | 0.7857 | 2671 |
| weight avg | 0.8272 | 0.7862 | 0.7845 | 2671 |

It is deduced (by precision column) that the 94 % of benign images and the 69 % of malignant images were classified correctly. Moreover, the model's accuracy on test set was 79 %.

## 6.3 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 a = $10^{-5}$. The model's confusion matrix found to be:

| | true | |
|---|---|---|
| | benign | malignant |
| **predicted as** benign | 1432 | 1069 |
| malignant | 60 | 110 |

The model's accuracy during the epochs of its validation is depicted in the diagram below. Model's maximum accuracy on validation set found to be approximately 64% in the third epoch.


SGD model accuracy

Furthermore, the per class statistics found to be:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| benign | 0.5725 | 0.9597 | 0.7172 | 1492 |
| malignant | 0.6470 | 0.0932 | 0.1630 | 1179 |
| accuracy | N/A | N/A | 0.5773 | 2671 |
| macro avg | 0.6098 | 0.5265 | 0.4401 | 2671 |
| weight avg | 0.6054 | 0.5773 | 0.4726 | 2671 |

By inspecting the precision column, it may be seen that the 57% of benign images and the 65 % of malignant images were classified correctly. Moreover, the model's accuracy on test set

was 57.73% (the highest recorded among the models that we tried).

## 6.4   Support Vector Machines

In SVM 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 Support Vector Machine model with an rbf kernel. The model's confusion matrix found to be:

| | true | |
|---|---|---|
| | benign | malignant |
| **predicted as** benign | 663 | 122 |
| malignant | 829 | 1057 |

The ability of making correct predictions over the validation set is shown below as the diagram of its accuracy during its training epochs. The model scored the biggest validation accuracy in the 2nd epoch and it was equal to 0,67 % .


SVM model accuracy
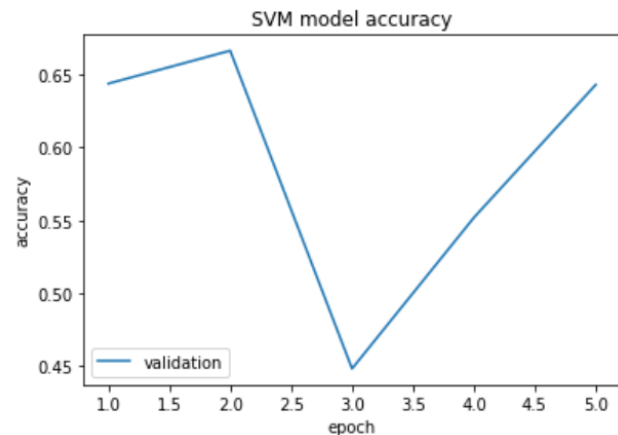
Furthermore, the per class statistics were:

|  | precision | recall | f1-score | support |
|---|---|---|---|---|
| benign | 0.8445 | 0.4443 | 0.5823 | 1492 |
| malignant | 0.5604 | 0.8965 | 0.6597 | 1179 |
| accuracy | N/A | N/A | 0.6439 | 2671 |
| macro avg | 0.7025 | 0.6704 | 0.6360 | 2671 |
| weight avg | 0.7191 | 0.6439 | 0.6297 | 2671 |

The model scored 64 % accuracy over the test set.

# 7   Conclusion

On our try to classify images of tumors as either benign, or malignant, we developed and configured machine learning models. We exploited two neural networks: MultiLayer Perceptron and Convolutional and two non-neural networks: Stochastic Gradient Descent and Support Vector Machine. The models overview is contained in the table that follows.

|  | MLP | CNN | SGD | SVM |
|---|---|---|---|---|
| Neurons (MLP); filter size (CNN) | (512, 512) | (32,32,32) | N/A | N/A |
| # Layers | 3 (3 neuronics) | 3 (2 conv + 1 neur) | N/A | N/A |
| # Epochs | 18 | 40 | **5** | 5 |
| Mean Epoch duration (s) | **218** | 862 | 243 | 609 |
| Accuracy (%) | 75 | **79** | 58 | 64 |

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

It is 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 needed the least mean time per epoch to be trained.

2. The first two models (MLP, CNN) achieved around 77% accuracy whereas the third (SGD) and fourth (SVM) achieved around 61 % accuracy. Moreover, based on the diffusion matrix of the third model, we conclude that it is worthless since there are many false positives. In other words, in most cases the model does not predict that the image depicts skin cancer.

As a concluding remark, we used the method "Early Stopping" on our neural networks so as to decide the number of epochs and prevent from overfitting.

Here is the validation  and test accuracy per model.

| % | MLP | CNN | SGD | SVM |
|---|---|---|---|---|
| valid accuracy | 74 | 78 | 64 | 66 |
| test accuracy | 75 | 79 | 58 | 64 |

Based on the above table we come to the concusion that the best model is the CNN.

Time elapsed for the final model (including data manipulation): **10 Hours**
Machine used: **8GB RAM,** CPU: **Intel(R) Core™ i7-5500U**

# 8   Members/Roles

Business Analyst: Vasiliki Karamesiou (B.Sc. in Economics)

Responsible for:

- Data Set Search
- Manipulate Data
- Data Investigation
- Configuration SGD
- Report Creation

Data Engineer/DevOps: Evangelos Kontaratos (B.Sc. in Computer Science)

Responsible for:

- Data Set Search

- Manipulate Data

- Configuration CNN

- Configuration MLP

- Report Creation

Data Engineer/DevOps: Erasmia Kornelatou (B.Sc. in Computer Science)

Responsible for:

- Data Set Search

- Manipulate Data

- Configuration CNN

- Configuration MLP

- Configuration SVM

- Report Creation

Business Analyst: Christos Schismenos (B.Sc. in Economics and Regional Development)

Responsible for:

- Data Set Search

- Manipulate Data

- Data Investigation

- Configuration SGD

- Report Creation

## 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:10.1561/2000000039.

[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]  Rezvantalab 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.

**Time Plan**

| 2020 Sept | | | | 2020 Oct | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S 26 | S 27 | M 28 | T 29 | W 30 | T 1 | F 2 | S 3 | S 4 | M 5 | T 6 | W 7 | T 8 | F 9 | S 10 | S 11 | M 12 | T 13 | W 14 | T 15 | F 16 | S 17 | S 18 |

Data Set search

ManipulateData

Data investigation

Configure MLP

Configure SGD

Configure CNN

Configure SVM

Report creation

MLP Reconfiguration

CNN Reconfiguration

Report Update

Presentation Creation

Team Work

Schismenos Christos, Karamesiou Vasiliki

Kontaratos Evangelos, Kornelatou Erasmia

Schismenos Christos, Kontaratos Evangelos

Karamesiou Vasiliki, Kornelatou Erasmia

Kornelatou Erasmia

**Contact Person Erasmia Kornelatou | Mobile:
6976936837 | email: f2821907@aueb.gr**

## Comments

The major difficulty we faced was during data loading on PCs memory. Images are treated as a 3 dimensional arrays (width x, height x, color layers). Thus, the attempt of loading the whole data set at once on memory, was causing program to crash due to exceeding memory limit. Besides, our goal was to make a generic technique that could manipulate a data set no matter of its size. Some of our time was spent in the proper feed of data to the models that we were interested in training. We decided to use the image data generator [3] functionality that was used to augment the initial data set.

This assignment was an amazing opportunity for us to expose ourselves to the wonders of artificial intelligence. We would also like to use this opportunity to thank our professors for connecting us with such inspiring and knowledgeable people, in addition to allowing us to explore these amazing new technologies on our own projects, which made us all a lot more invested and excited for the assignment.