Convolutional Neural Networks versus Feature Extraction

Bruno Junceiro Sousa
Instituto Superior Técnico
Universidade de Lisboa
Lisboa, Portugal
bruno.junceiro.sousa@tecnico.ulisboa.pt

***Abstract*—Although Convolutional Neural Networks revolutionized the Deep Learning and computer vision fields, state-of-the-art models have been suffering from some problems, due to their increasing depth. These problems are: requirement of enormous labeled datasets for their learning; requirement of huge computational power to process those amounts of data; degradation of their generalization ability the deeper they become. With the purpose of solving those problems, we present a classification architecture which focus on the extraction of the color, texture and shape information from images, using computer vision techniques. This information will be aggregated based on their feature type (modality) and used to train independent models. Each modality will be learned by a three different models, with the intention of improving the generalization of the proposed solution. We chose the best performing model to represent the three modalities. From these models, we will use the resulting Softmax probabilities assign to each modality and fuse them using the Dempster-Shafer's rule of combination, as the multimodal fusion approach. To train and experiment on the proposed solution we used the Fruits 360 dataset and evaluated the performance on said experiments with the loss and accuracy metrics. The implemented multimodal classification model was compared with two baseline CNNs, which were trained using the same configurations: batch size, epochs, activation and loss functions. The baseline models achieved accuracies of 93.46% and 93.34%, while the proposed solution achieved 94.80% for the fusion of the color, texture and shape modalities. We were able to improve over this result, using the Dempster-Shafer's rule of combination with only the color and texture modalities, which resulted in a accuracy of 97.40%.**

***Index Terms*—Feature Extraction, Convolutional Neural Networks, Image Classification Tasks, Multimodal Fusion, Dempster-Shafer Theory, Deep Learning**

I. Introduction

Deep Learning neural networks, specifically Convolutional Neural Networks, are major contributors to the current state of the computer vision field. Since the first backpropagated CNN [1], we have been solving image recognition tasks of increasing complexity.

Starting with the Neocognitron [2], in 1980, models motivated by the notions presented by Hubel and Wiesel [3] [4], such as the receptive fields in the visual cortex of mammalian brains, have been constantly improving the field of digital image processing.

Some of these models, like GoogLeNet [5] and ResNet [6], were groundbreaking to state-of-the-art CNNs. Due to these networks, we are currently able to create Convolutional Neural Networks of enormous depth, capable of learning datasets with millions of samples distributed over thousands of labels.

Although all the important innovations introduced by the mentioned models, modern Convolutional Neural Networks are facing some shortcomings. Generally, the depth of a neural network is associated with the amount of labeled samples and computational power required for it to successfully learn. So, as expected, the continuous depth increase of CNNs is starting to cause problems, such as lack of labeled datasets with the amount of samples required and hardware limitations. The computational power required to train state-of-the-art CNNs worsened with each increase of the CNN's depth. This problem may not be of relevance for the creators of the GoogLeNet and ResNet (Google and Microsoft, respectively), because these companies are in the vanguard of computer science and, consequently, have the necessary resources to train these CNN's at their disposal. However, smaller companies and research teams do not have those same resources, which limits their efficient use of networks with this increasing depth.

Additionally, despite deeper CNNs being able to learn more complex features, an excessive depth can cause the degradation of the network's generalization ability.

A. Objectives and Proposed Solution

Considering the aforementioned problems, a possible solution would be to resort to computer vision techniques in conjunction with simpler models, such as shallow Neural Networks, in order to extract only certain features from the datasets, making the data samples smaller and easier to process by the model.

We propose a classification architecture, composed of three modules: Feature Extraction, Feature Learning and Multimodal Fusion; that intends to be a solution to to the problem of the computational power required to train deeper CNNs.

The proposed model aims to use simpler Neural Network models to learn feature vectors of considerably lower dimensionality than the original images.

The Feature Extraction module is where the images will be processed and their respective features extracted, through use of computer vision techniques, and organized into feature vectors, according to the modality which they represent. The features we intend to extract from the images are the color,

texture and shape information. The color information will be extracted using color histograms, to represent the pixel intensity for each color channel, and will be represented in a feature vector that consists of a normalized concatenation of the three color histograms. The texture information will be extracted computing the image gradient and will be represented using a Histograms of Oriented Gradients, which consists of an aggregation of multiple local gradient histograms computed for each region of the image. The shape information will be extracted using several methods, such as edge and corner detection. This information will the be combined in a feature vector we designated as Shape Map.

On the Feature Learning module, we will train several Neural Network models for each modality extracted on the previous module. These Neural Networks will be trained over the same circumstances and we will choose the best performing model for each modality. The reason for having Neural Networks focus specifically on learning only one feature type is the fact that this will greatly increase the generalization of the each model. This approach was inspired by how the human eye has different types of cells to process color (Cones) and light intensities (Rods).

After choosing the model that will represent the Color, Texture and Shape modalities, we will use those models to predict the labels of the test set, in order to create probabilistic vectors with a dimension equal to the number of labels present in the dataset. These probabilistic vectors will then be combined in the Multimodal Fusion module, using the Dempster-Shafer's theory of evidence [7]. This multimodal fusion technique will use the probabilistic vectors as mass functions, which will be combined resorting to the Dempster-Shafer's rule of combination. From the final combined mass functions, we will finally obtain the predicted labels for the samples in the test set.

Considering the generalization problem, we will also experiment on the Multimodal Fusion module with combinations of only two modalities, leaving one out at a time. With to this approach, we can evaluate if our solution is capable of generalize without considering all the extracted features.

We intend to also create some Convolutional Neural Networks, with the purpose of having a baseline to compare the results obtained through the Multimodal Fusion classification.

B. Evaluation Methodology and Dataset

To evaluate both the proposed solution and the Convolutional Neural Networks used as a baseline, we will resort to metrics such as accuracy and loss. These metrics are of great importance to analyze the training experiments that we will perform on these models

For the Multimodal Fusion module we will also use the precision and recall metrics, which allow a better understanding of how the models are performing, at a label level.

The implementation of the proposed classification architecture, and respective experiments, will be supported by the use of the Fruits 360 dataset [8], which is composed of a total of 90,380 images distributed over 113 labels.

II. Related Work

The models presented in Table I are considered as milestones in the Convolutional Neural Networks field, because of the important innovations each of them introduced:

- **LeNet-5** [9] is considered as the start of using the convolution operation integrated in a Neural Network;
- **AlexNet** [19] introduced the use of the ReLU Activation Function of the network neural unit and the Max Pooling downsampling approach. It was also the first CNN to parallelize the training process between two GPUs;
- **GoogLeNet** [5] was one of the first major depth increases in a CNN. This network introduced the concept of Inception Modules and Auxiliary Classifiers, in order to solve the vanishing gradient problem, which is inherent to the increased depth of the model;
- **ResNet** [6] surpassed all the previous CNNs, with a depth of 152 layers, which later was expanded to a 1001-layer version, to learn over both the CIFAR-10 and CIFAR-100 datasets. This network introduced a new approach for solving the vanishing gradient problem, which consist of using Residual Blocks and Skip Connections.

Model	Year	Layers	Dataset
LeNet-5	1998	8	MNIST
AlexNet	2012	12	ImageNet
GoogleNet	2014	22	ImageNet
ResNet	2015	152	ImageNet
ResNet	2016	200	ImageNet
ResNet	2016	1001	CIFAR-10 CIFAR-100

TABLE I
EVOLUTION OF THE CNNS MILESTONES.

Considering the these milestones, we can easily infer that the tendency in state-of-the-art CNNs has been the increase of the structures depth. Observing the Table I, we can see that in less than 20 years we went from an 8-layer network (1998 LeCun's LeNet-5) to a 1001-layer network (2016 Microsoft's ResNet), which represents an increase of over 125 times.

Each new milestone introduced innovations that allowed models to learn more complex datasets. Those innovations made it possible to move from learning a dataset of simple grayscaled (1-channel color space) images, belonging to one of 10 possible labels (MNIST), to a dataset of much more complex colored (3-channel color space) images, belonging to one of 100 different labels (CIFAR-100).

Although, the deeper networks, such as GoogLeNet and ResNet, managed to create solutions to fix the vanishing gradient problem, which is inherently aggravated the deeper the structure of the model is, another problem emerged: the computational power required to train such models worsened with each increase of the CNN's depth.

The computational power may not be an urgent problem to the creators of GoogLeNet and ResNet (Google and Microsoft, respectively), because these companies are in the vanguard

of computer science and, consequently, have the necessary resources to train these CNN's at their disposal. It is highly probable that smaller companies or even research centers do not have a fraction of those resources, which limits their efficient use of networks with this increasing depth.

A possible solution may be to resort to computer vision techniques in conjunction with simpler models, such as ANNs, in order to extract only certain features from the datasets, making the data samples smaller and easier to process by the model. This thesis intends to ascertain if this is a viable solution to the problem of the computational power required to train deeper CNNs.

III. Implementation

Considering the problems of generalization and shortage of large labeled datasets, we propose a solution with the objectives of learning features present in a dataset of images and solving the problem of the computational power required to train deeper CNNs.

The proposed solution consists of a classification architecture with the following three modules:

- **Feature Extraction** - the features (color, texture and shape) are extracted from the images, resorting to computer vision techniques;
- **Feature Learning** - each feature category (color, texture and shape) will be learned separately, by independent Neural Network models. This module will have three different machine learning models, each responsible of learning one of the modalities;
- **Multimodal Fusion** - the probabilistic outputs, which result from each model of the Feature Learning module, are combined using a Multimodal Fusion technique, in order to predict the final label that should be assigned to each image. Since this module is responsible for the label prediction for the images, it will only be used on the images which belong to the test set.

The idea of learning different modalities independently was inspired by the cells found in the human eye's retina [10]:

- **Rods** - cells that react to light, detecting different intensities. Intensity variations help us recognize different shapes and even textures, even when we are under dim lights or outside at night;
- **Cones** - cells that react to light, detecting the wavelengths that we interpret as the different colors from the visible spectrum. Our retina has three types of cones, whose purpose is to independently capture each of the colors red (R), green (G) and blue (B).

Using independent models for each feature category,also known as modality, allows us to avoid the problem of certain features negatively impacting the others. This problem usually arises between the light intensity and color, because the former can induce humans to mistakenly identify the color of certain objects, depending on how intense is the light used to illuminate those objects.

Additionally, using this approach we aim to construct models capable of better generalizations, since each one of them will exclusively focus on learning a single modality.

In the Multimodal Fusion module, we will experiment with the omission of one of the modalities, in order to evaluate if the proposed solution is able to correctly classify the images, even in these conditions.

The proposed classification architecture will be compared to Convolutional Neural Networks, which will be trained in the same circumstances: dataset, activation functions, loss function and epochs. These CNNs will serve as a initial baseline.

A. Dataset

For the training of the baseline CNNs and the models from the Feature Learning module, we used the Fruits 360 dataset [8]. This dataset consists of a total of 90,380 images, each with a size of 100x100 pixels, distributed over 131 labels. Some of the labels are similar, only being differentiated by a number at the end of the label. Contrary to labels such as "Apple Braeburn" and "Apple Pink Lady", which represent different varieties of apples, the labels "Apple Golden 1", "Apple Golden 2" and "Apple Golden 3" represent the same variety of apple. For that reason, we merged labels in those situations, resulting in a decrease from 131 to 113 labels.

The dataset is separated between a train and test sets: 67,692 images on the train set and 22,688 images on the test set. This represents a train-test split of approximately 75%-25%. To create a validation set, which will be used to validate the training process of the classification models, we applied a split of 75%-25% to the train set, preserving the same proportions of samples from each label as observed in the original train set. This resulted in a new train set with 50,769 images and a validation set with the remaining 16,923 images.

B. Baseline

We chose two models to serve as a baseline for the proposed solution. Both models are Convolutional Neural Networks which we previously created to learn a subset of the Fruits 360 dataset. This subset consisted of 14,419 images distributed between 4 super-labels: Apples, Lemons, Oranges and Pears.

These two models have some similarities, such as: only having one Convolutional Layer; using Softmax as the activation function on the Output Layer; and using the Cross-Entropy Loss Function on the backpropagation algorithm. The main structural differences between them is that the enquoteBaseline Beta model has a Pooling Layer, which performs the Max-Pooling downsampling, and uses the Dropout regularization technique in two sections of the structure.

In order to serve as a baseline, both these models were trained over the complete dataset, without resizing the images and following the train-validation-test split defined in the Section III-B.

The only pre-processing done to the images was the normalization of their values. Considering the images consist of three color channels, each encoded in 8 bits, we normalized each image by dividing its values by 255, which is the maximum value of color intensity represented by that encoding.

C. Feature Extraction - Color

To represent this modality, we chose to extract the color feature using color histograms. The images of the Fruits 360 dataset are represented in the RGB color space, which means that each image is composed of three channels: Red, Green and Blue.

Firstly, we separated each image into the three respective color channels. This process transformed the images from 3-channel matrices to three individual 1-channel matrices.

After separating the color channels, we computed a color histogram for each of them. The color histogram represents the quantity of pixels, from a certain image, that are assign to each level of color intensity

Since the images from this dataset are stored using an 24 bit encoding, each channel uses 8 bits to represent the color intensity. This means that our color histograms will use a range of bins from 0 to 255, having 256 (2^8) levels of intensity.

When comparing the color histograms of each color channel, we noticed that all three color channels present the highest pixel counts for the intensity levels above 250. This is justified by all images presenting a completely white background. We decided to remove the background from the images using a mask created from each image, which would be used to ignore the pixels from the background area of the image, during the computation of the color histograms.

In order to create the masks, these are the steps we followed:

- **Color Space Conversion** - in this step, we converted the images from the RGB to the Grayscale color space. This transformation results in a single channel image, whose values are computed with (1);
- **Image Thresholding** - in this step, we applied the Binary Inverse Thresholding approach to the grayscale images [11]. This thresholding approach converts pixel intensities to 0, if their value is above the defined threshold, or to 1, otherwise. Considering we intended to remove the white background, the value we chose for the threshold was 250. This transformation results in a binary image;
- **Morphological Transformation** - observing the results from the Image Thresholding step, we noticed that the images had some noise around the fruits' silhouette. This step aims to clear that noise from the binary images, through use of the Opening Morphological Operator [12]. This Morphological Operator uses a structured element (also known as kernel) to remove small areas of activated pixels (with value 1) which are situated around bigger areas of activated pixels, associated with objects present in the image. The kernel is used across the image, computing convolutions which result in the activation or deactivation of the pixels. We used the kernel in (2) for the Opening operation.

$$\text{Grayscale} = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{1}$$

$$Kernel_{Opening} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2}$$

Having created the masks for every image in the dataset, we computed the new color histograms. These updated color histograms were computed using the masks to decide which pixels should be considered. The decision consisted of if a pixel falls inside the activated region of the mask, then that pixel is used for the histogram calculation, otherwise it will be ignored.

Before using these histograms to learn the color modality, first we had to normalize their values, to better fit the Neural Network models. We considered to normalization techniques: Probabilistic Normalization and Min-Max Normalization. The former transforms the histograms into a probabilistic model, in which its values are divided by the total amount of pixels. On the other hand, the latter technique maps the histogram values into a new range, defined by a given minimum and maximum values.

When comparing these normalization techniques, we concluded that, although both were capable of achieving the objective of normalizing the histogram values into a new range between 0 and 1, the Probabilistic Normalization transformed most of the values into small probabilities, very close to the minimum value (0).

Having decided on the normalization technique, we created the color feature vectors for each data sample. These features vectors are a concatenation of the normalized color histograms and have a size of 768, which resulted from the sum of the previous three color histograms' sizes (256).

We built three Neural Networks to train over the extracted color feature vectors. These models have the names "Color Alpha", "Color Beta" and "Color Gama". Both the "Color Alpha" and "Color Beta" models only have one hidden layer, being differentiated by the number of hidden units. The latter model has double the hidden units (256) of the "Color Alpha" model (128). The "Color Gama" model has two hidden layers: the first with 384 units and the second with 192 units.

D. Feature Extraction - Texture

To represent this modality, we chose to extract the texture feature using Histograms of Oriented Gradients (HOG) [13].

Firstly, we transformed the images from the RGB color space to the Grayscale color space. This is a required pre-processing to compute the image gradient.

Having the images in grayscale, we computed the image gradient for all samples, using the Sobel operator [14]. The gradient of an image is computed as the first-degree derivative of a 2-dimensional continuous function. The Sobel operator adds to this notion, computing the derivative through a convolution operation between the image and the respective Sobel kernels. The kernels used in the convolution operation depend on the kernel size chosen in the operator. We experimented with the kernel sizes 3 and 5, which were assigned to the kernels (3) and (4), respectively.

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3}$$

$$M_x = \begin{bmatrix} 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \\ 4 & 2 & 0 & -2 & -4 \\ 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \end{bmatrix} \quad M_y = \begin{bmatrix} 2 & 2 & 4 & 2 & 2 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -2 & -1 & -1 \\ -2 & -2 & -2 & -2 & -2 \end{bmatrix} \tag{4}$$

From the application of the Sobel operator to an image, we obtain two matrices: the vertical and horizontal image gradients.

Comparing the results of both Sobel kernel sizes, we noticed that using a kernel size of 5, associated with the kernels (4), detects a lot more noise surrounding the edges found on the image gradients. Using a kernel size of 3 (3), we can obtain the same edges with significantly less noise. Although, we can still detect some nuances of textures inside the object. After comparing the results from both kernel sizes, we chose to use the size 3 for the kernels of the Sobel operator.

The next step is to calculate the gradient magnitude and orientation, for each pixel in the image. To calculate the values of magnitude and orientation, we use the following formulas:

$$Magnitude = \sqrt{Grad_x^2 + Grad_y^2} \tag{5}$$

$$Orientation = \left| \tan^{-1} \left(\frac{Grad_y}{Grad_x]} \right) \right| \tag{6}$$

Using the vertical and horizontal gradients to compute the magnitudes (5) and orientations (6) of the gradient, results in two new matrices, with the same size of the original image. From the gradient magnitude matrix, we verified that the edges of the object have the highest magnitude values, while the pixels from the background have a zero value magnitude. Observing the gradient orientation matrix we could also verify that the highest angle values are distributed across the silhouette of the objects (fruits), considering its roundness.

With the magnitude and orientation matrices, we can finally create the Histogram of Oriented Gradients. The HOG is a computer vision technique used to represent texture information and is created following the steps below:

1) We start by dividing the magnitude and orientation matrices into blocks. These blocks can have a custom size, and for the current example we will consider a size of 10x10, since it is easier to visualize for an image with size 100x100. This results in 100 blocks, since we have a length of 10 blocks both vertically and horizontally;
2) For each block, we create an oriented gradient histogram. The bins for this histogram represent the orientations and the values represent the sum of magnitudes related with those orientations. The number of bins for the histogram can have a custom value, which will affect the step size between bins. Considering the bins represent the orientation, which ranges from 0° to 180°, we will use 9 bins for the current example. With a 9-bin histogram, the steps between bins will be of 20°;
3) Since this is not a common histogram, like the color histograms explored in Section III-C, we have to consider that each bin is selected by the orientation value and the value that is added to the chosen bin is selected from the magnitude matrix. This means that, if the gradient orientation of a certain pixel is not the exact value attributed to one of the bins, the magnitude value associated with that pixel will be proportionally split between two bins. Considering the example of a magnitude value of 60 associated with a orientation of 170°, that magnitude will be split 50-50% between the bins 160 and 0 (30 for each), since 170° is at a distance of 10° from both 160° and 180°. Even though the orientation value is between 160 and 180, we are only considering angles from 0° to 180°, which means the angle will wrap around, making 0° and 180° equivalent;
4) Having computed the histograms for each of the 100 blocks, we will have to normalize the blocks, to prevent possible lighting variations captured by the gradient of the image. To normalize the blocks we use a window of size 2x2 blocks which will slide across the image, concatenating the 4 histograms contained inside the window. The concatenation of the histograms will result in a vector with size 36x1 (4 histograms with 9 bins each). This vector will then be normalized with the L2-norm (7). The variable ε in (7) is a small value added to prevent zero division;
5) Finally, the HOG is created by concatenating all the normalized vectors, which for the given example would have a dimension of 2916x1.

$$\text{L2-norm: } \frac{\vec{v}}{\sqrt{\|\vec{v}\|^2] + \varepsilon}} \tag{7}$$

We built three Neural Networks to train over the extracted texture feature vectors. These models have the names "Texture Alpha", "Texture Beta" and "Texture Gama". Both the "Texture Alpha" and "Texture Beta" models only have one hidden layer, being differentiated by the number of hidden units. The latter model has more 128 hidden units (384) than the "Texture Alpha" model (256). The "Texture Gama" model has two hidden layers: the first with 384 units and the second with 192 units; similarly to the structure of the "Color Gama" model.

E. Feature Extraction - Shape

To represent this modality, we chose to extract the shape feature using a Shape Map. The Shape Map is our proposed solution for representing shape information and consists of an aggregation of multiple features associated with the shape feature. We propose a Shape Map which combines the silhouette of the object, the edges detected with the Canny algorithm [15] and the corners detected with both the Harris [16] and Shi-Tomasi [17] algorithms.

Similarly to the texture extraction, we started by transforming the images from the RGB color space to the Grayscale color space, since this is a required pre-processing for the extraction of the shape features that we chose to explore.

In order to detect the objects silhouette, we resorted to the previously implemented approach of removing the image background (Section III-C). However, instead of creating the

mask to remove the white background from the image, we use the same mask as a representation of the fruits silhouette. The silhouette is a binary image composed of active pixels (value 1), which represent the object, and inactive pixels (value 0), which represent the area outside of the detected object.

When detecting edges from an image, we have to consider that this process is greatly susceptible to noise in the image. To prevent this problem, we removed possible noise in the images with a 3x3 Gaussian filter. The 3x3 kernel used on this filter is created with the Gaussian function $G(x, y)$ (8).

$$G(x,y) = \alpha e^{\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}} \tag{8}$$

In this Gaussian function, α is a scale factor chosen to guarantee that (9) is true.

$$\sum_{x,y} G(x,y) = 1 \tag{9}$$

After filtering the images, we used Canny's Edge Detector to extract the edge information. This edge detection algorithm processes the images through the following steps:

1) **Image Gradient** - the algorithm computes both the horizontal and vertical image gradients with a Sobel operator, and from those gradient matrices computes the gradient magnitudes and orientations, similarly to what we did to extract the texture information (Section III-D);
2) **Non-maximum Suppression** - the algorithm scans the image removing pixels which do not belong to one of the edges. This is accomplished by checking for each pixel if its magnitude value is a local maximum among the neighbor pixels that follow the same gradient orientation. This step results in a binary image with all the detected edges;
3) **Thresholding through Hysteresis** - in this step, the algorithm filters the edges detected in the previous step, resorting to a low-threshold and a high-threshold. For each detected edge pixel, the algorithm will check where the corresponding magnitude value is situated in comparison to the defined threshold values. If the magnitude value is higher than the high-threshold, that pixel is classified as a true edge. On the other hand, if the magnitude value is lower than the low-threshold, the pixel is discarded. For the last possibility, if the magnitude of an edge pixel lies between the two thresholds, the pixel will only be classified as an edge if it is connected to an already confirmed edge.

After experimenting with the values for the Canny Edge Detector's thresholds, we decided to use a low-threshold of 30 and a high-threshold of 120, which allowed Canny's algorithm to detect both the outside contour and some edges inside of the fruits.

The first corner detecting algorithm we chose was the Harris Corner Detector. This algorithm detects edges and corners in an image, by scanning the image, with a window function $w(x, y)$, and classifying the region inside that window as a corner, edge or flat. This classification will depend on the value of a score R.

The Harris detector computes the image gradient with the Sobel operator, which will be then used to compute the R score. This score is calculated using the formula in (10).

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{10}$$

The variables λ_1 and λ_2 in the score R formula represent the eigenvalues of the matrix M, calculated by the equation (11).

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} Grad_xGrad_x & Grad_xGrad_y \\ Grad_xGrad_y & Grad_yGrad_y \end{bmatrix} \tag{11}$$

According to the rules of the Harris Corner Detector, each region of the image is classified as:

- **Corner** - large R score, which occurs when both λ_1 and λ_2 are large values and $\lambda_1 \sim \lambda_2$;
- **Edge** - $R < 0$, which occurs when $\lambda_1 >> \lambda_2$ or $\lambda_1 << \lambda_2$;
- **Flat** - $|R|$ is small, which occurs when both λ_1 and λ_2 are small values.

The configuration chosen for the Harris Corner Detector algorithm was: window with size 2 and k factor of 0.025.

For the detection and extraction of corners, we also resorted to the Shi-Tomasi Corner Detection algorithm. In this algorithm, J. Shi and C. Tomasi proposed a small modification to the Harris Corner Detector, which ended up improving the results achieve by that algorithm. The proposed modification consisted in changing the scoring function R from (10) to (12).

$$R = \min(\lambda_1, \lambda_2) \tag{12}$$

With the presented modification to the scoring function R, this algorithm also adapted the rules used to classify each region of the image. The new rules are as follows:

- **Corner** - both λ_1 and λ_2 are above a specified minimum value (λ_{min});
- **Edge** - only one of the eigenvalues λ_1 and λ_2 is above λ_{min};
- **Edge** - both λ_1 and λ_2 are below λ_{min};

The configuration chosen for the Shi-Tomasi Corner Detector algorithm was: λ_{min} with value 0.01.

Comparing the results from both Corner Detectors, we noticed that the Harris algorithm finds more corners along the object contour, while the Shi-Tomasi algorithm is able to find corners inside the object's area.

As aforementioned, we propose a Shape Map, that converges various types of shape information, as the feature vector for the Shape Modality. Our Shape Map will consist of a weighted aggregation of the binary images that resulted from the previously presented methods of extracting shape information.

The weight values were chosen within the range [0, 255], which will result in a grayscale image. The weights attributed to each type of shape information are as follows:

- **Silhouette** - 64;
- **Edges** - 128;
- **Harris Corners** - 224;
- **Shi-Tomasi Corners** - 255.

The weights were attributed considering the relevance of each type of shape information. Corners detected by the Shi-Tomasi detector will have a higher weight, in comparison to Harris' corners, since the former is an improvement of the latter algorithm.

The aggregation process will always choose the highest weight possible for each pixel, that is, in a scenario where a pixel is active, for example, in both the Edge and Harris Corners binary images, it will be associated with the weight of the latter. Pixels that are inactive in all four shape binary images will have a weight of 0.

Finally, we normalize and resize the Shape Maps. The normalization approach used consists of dividing the values by 255, adjusting their range to [0,1]. We resize the Shape Maps, by a scale of 50%, as a means of dimensionality reduction. The resized maps have a size of 50x50.

We built three shallow Convolutional Neural Networks to train over the extracted Shape Maps. These models have the names "Shape Alpha", "Shape Beta" and "Shape Gama". All three models have one Convolutional layer, with a kernel size (10,10). The difference between the models is the number of defined filters: "Shape Alpha", "Shape Beta" and "Shape Gama" have 8, 16 and 32 filters, respectively.

F. Multimodal Fusion

Multimodal Fusion is the combination of the results obtained from various independent modalities. Being an image classification task, the modalities we defined are associated with the color, texture and shape information, extracted from a dataset of images.

The Multimodal Fusion can be performed through several different methods. For example, We could simply use a weighted-sum, where the weights represent the contribution percentage that each modality has on the final value.

We decided to rely on the Dempster-Shafer's theory of evidence to combine the Softmax probabilities, that result from the classification of the test set by each Modality model.

With the Dempster-Shafer's theory of evidence it is possible to associate measures of uncertainty with sets of hypotheses, when the individual hypothesis are imprecise of unknown [18]. The Dempster-Shafer's rule of combination derives common shared belief between multiple sources and ignores all the conflicting beliefs [7].

In the context of this work, our hypothesis will be if either that is the label associated with the image. All possible mutually exclusive hypothesis are contained in a frame of discernment θ. The possible combinations of hypothesis, including the empty set $\emptyset$, are represented by 2^θ.

The main advantage of using the Dempster-Shafer's theory is that it supports the specification of a degree of uncertainty, instead of forcing the hypotheses to have probabilities that add to unity, just like a traditional probability theory. However, we will not need this advantage, since the results from our modalities are probabilities calculated by the Softmax Activation Function.

The Dempster-Shafer's theory allows the definition of a belief mass function m. This belief mass function is applied to each element contained in θ, following these three requirements:

- $m : 2^\theta \rightarrow [0, 1]$;
- $m(\emptyset) = 0$;
- $\sum_{A \epsilon 2^\theta} m(A) = 1$.

The probability of a label A being perceived as correct, for the image processed by a modality, is given by the confidence interval $[Belief(A), Plausibility(A)]$.

Belief represents the lower bound of the confidence interval and is defined as being the total evidence that supports the hypothesis. The belief is calculated as the sum of all the masses of the subsets associated with the set A (13).

$$Belief(A) = \sum_{B|B \subseteq A} m(B) \tag{13}$$

Similarly, Plausibility is the upper bound of the confidence interval and is defined as being the sum of all the masses of the set B, that intersects the set of interest A (14).

$$Plausibility(A) = \sum_{B|B \cap A \neq \emptyset} m(B) \tag{14}$$

The Dempster-Shafer theory provides a combination rule (15) to combine evidences detected by two different modalities S_1 and S_2.

$$(m_{S_1} \otimes m_{S_2})(A) = \frac{1}{1-K} \sum_{B \cap C = A \neq \emptyset} m_{S_1}(B) m_{S_2}(C) \tag{15}$$

The K in the formula 15 measures the conflict between the two modalities and is calculated by the equation 16.

$$K = \sum_{B \cap C = \emptyset} m_{S_1}(B) m_{S_2}(C) \tag{16}$$

In the context of our work, the mass functions for each modality represent the probability of each label being correct for a certain image. In other words, the mass functions will be the probability values given by each Modality model.

Considering we have three modalities, one for each type of features extracted (color, texture and shape), we had to apply the Dempster-Shafer's combination rule in two iterations. On the first iteration, we simply used the mass functions of two modalities. For the second iteration, applied the combination rule between the mass functions from the remaining modality and the results from the first iteration. This results in combined mass functions that represent the fusion of the Color, Texture and Shape Modalities.

Lastly, the Multimodal Fusion module will predict the final labels for the test set. This prediction is done by choosing, for each image, the label assigned to the highest value from the combined mass functions.

IV. Experimental Evaluation

For the experiments of training the baseline CNNs and the Modality models, we used the same training configurations: train over 10 epochs, with a batch size of 32 and Cross-Entropy as the models loss function.

A. Training and Testing the Modality Models

We decided which model will represent each modality, accordingly to their performance learning the feature vectors with the extracted information.

The models chosen to represent each modality were:

- **Color Modality** - "Color Gama" model, trained with a batch size of 32 and applying a background removing mask to the images before computing the color histograms;
- **Texture Modality** - "Texture Gama" model, trained with a batch size of 32 and with the HOG parameters: block size 20x20 and bin step 15°;
- **Shape Modality** - "Shape Gama" model, trained with a batch size of 32 and with the resized Shape Maps.

These were the models chosen to classify the test set. Testing the models serves for more than simply obtaining the test accuracies. The main reason to process the test set is so that the models can generate the Softmax probabilities for the data samples.

When using the Softmax Activation Function, the models prediction consists of a probability vector. Each position of this vector represents the predicted probability of a sample being assigned to the corresponding label.

Model	Accuracy		
	Train	Validation	Test
Color Modality	98.37%	94.23%	95.05%
Texture Modality	92.44%	75.64%	73.81%
Shape Modality	99.75%	68.87%	69.96%

TABLE II
SUMMARY OF THE ACCURACY VALUES FOR EACH MODALITY MODEL.

The Table II summarizes the accuracy values that each Modality model achieve for the train, validation and test sets. The test accuracy values are in line with the validation accuracies. Both the Color and Shape Modality models had a slight increase from the validation accuracy, while the Texture Modality model decreased less than 2% for the test set.

B. Analyzing the Multimodal Fusion Results

Using the Dempster-Shafer's rule of combination on the Softmax prediction probabilities, produced from our modalities' classification of the test set, resulted in a combined accuracy value of 94.80%.

The Multimodal Fusion achieved a significantly better accuracy than both the Texture and Shape Modalities. In comparison with the Color Modality's test accuracy, the Multimodal Fusion achieved almost the same accuracy, with a small difference of 0.25%.

Analyzing from a label perspective, we extracted the following statistics:

- 61 labels, out of 113, have Precision = 100%;
- 1 label, out of 113, has Precision $< 60\%$;
- 66 labels, out of 113, have Recall = 100%;
- 2 labels, out of 113, have Recall $< 60\%$.

From these statistics we can conclude that more than 50% of the labels are being perfectly assigned, while only 3 out of the 113 labels present lower precision or recall values.

Although we achieved a good accuracy (94.80%) using the Dempster-Shafer's theory of evidence to combine our modalities, we decided to experiment using the same approach but only considering two out of the three modalities at a time.

Modalities			Accuracy
Color	Texture	Shape	
X	X	X	94.80%
X	X		97.40%
X		X	94.20%
	X	X	81.58%

TABLE III
SUMMARY OF THE MULTIMODAL FUSION EXPERIMENTS.

The Table III summarizes the resulting accuracies obtained for each possible combination of the Color, Texture and Shape Modalities.

From these results, it is clear that, for the Fruits 360 dataset, the most significant modality is the Color, since the Multimodal Fusion module achieves the lowest accuracy when combining only the Texture and Shape Modalities. Consequently, the least relevant modality for this dataset seems to be the Shape.

The combination of the Color and Texture Modalities achieved an accuracy of 97.40%, showing an increase of slightly over 2.5% in comparison with the initial experiment of fusing all three modalities. Using the Color and Texture Modalities also improved the precision and recall metrics:

- 85 labels, out of 113, have Precision = 100%;
- All of the labels have Precision $> 60\%$;
- 85 labels, out of 113, have Recall = 100%;
- 2 labels, out of 113, have Recall $< 60\%$.

These statistics show that over 75% of the labels are being perfectly assigned, having both perfect precision and recall, which represent an improvement of 25% over the precision and classification results for the first experiment. This combination also improved the overall values for precision metric, since none of the labels achieved a precision lower than 60%.

Analyzing the results, we can conclude that for the Fruits 360 dataset the best approach would be to use the proposed classification architecture extracting only the color and texture information from the images.

C. Comparison with other Models

Having concluded all the experiments and obtained the final values for the evaluation metrics, we compared the

performance of our implementation with the two baseline CNNs. In the Table IV, we present the accuracies for the baseline models and for the best modality combinations from the application of the Dempster-Shafer's combination rule. All the models being compared were trained in the same scenario: batch size 32 and 10 epochs.

Observing Table IV, we can see that both baseline CNNs underperformed, in comparison with the results for the Multimodal Fusion. Both the baseline models and the implemented classification architecture were trained in the same circumstances, as previously mentioned, and used shallow Neural Networks (even the CNNs).

Model	Accuracy
Baseline Alpha	93.46%
Baseline Beta	93.34%
Dempster-Shafer [Color, Texture, Shape]	94.80%
Dempster-Shafer [Color, Texture]	97.40%

TABLE IV
COMPARISON BETWEEN THE FINAL RESULTS AND THE BASELINE MODELS.

One aspect that is important to refer is that our implementation achieved more 4% accuracy using only two out of the three modalities extracted from the images. The Color and Texture Modalities combined have a dimensionality of 1,536 (768 from the color histograms and 768 from the HOG), while using the images represents a dimensionality of 30,000 (size 100x100 in a 3-channel color space). Considering all this, we can conclude that in the same train circumstances, the implemented solution achieved better results, while resorting to smaller samples and simpler NNs than the baselines.

We decided to compare our results with another model trained with the same dataset (Fruits 360): Muresan-Oltean's model [8]. Muresan and Oltean are the creators of the Fruits 360 dataset. They presented the dataset and consequently created a model to learn from those images. Their model consists of a 12-layers Convolutional Neural Network, with 4 pairs of Convolutional and MaxPooling layers. They trained this model for 25 epochs with a batch size of 50 samples, on 3 different scenarios, obtaining the following test accuracies for each:

- **Grayscale color space** - 95.25%;
- **RGB color space** - 98.66%;
- **HSV color space** - 96.09%.

Comparing the results from both our implementation and the Muresan-Oltean's model, its noticeable that our Multimodal Fusion classifier achieved a better accuracy than their Grayscale and HSV experiments. However, their RGB experiment achieved an accuracy 1.26% higher than our best experiment (Multimodal Fusion of the Color and Texture Modalities).

It seems that our solution slightly underperformed in comparison. Although, if we consider the complexity of Muresan-Oltean's model and the fact that they trained their model for 25 epochs, while all models from our solution were trained only for 10 epochs, we can claim that a difference of 1.26% is an acceptable trade-off for having both simpler models and data samples, two factors which have a great impact in reducing the computational power required to train Deep Learning models.

V. CONCLUSIONS

A new branch of Machine Learning models, designated as connectionist models, began with Rosenblatt's proposal of the Perceptron algorithm, which was inspired by the artificial neuron model previously presented by McCulloch and Pitts. Following important innovations, inspired by the Perceptron and its simulation of a neuron, models known as Artificial Neural Networks were created.

Some years later, Hubel and Wiesel discovered that the mammal brain has receptive fields inside its visual cortex. This was another important breakthrough, which launched models, such as Fukushima's Neocognitron, that envisioned the perception and processing of optical stimuli similarly to the human eye.

The notions presented by the those models and the introduction of the backpropagation algorithm, as the main learning approach used by neural networks, led to the creation of the first Convolutional Neural Network, proposed by LeCun.

Following several improvements over the first proposed model, CNNs have become one of the most widely used approaches to solve image recognition tasks.

Modern CNNs take advantage of many recently proposed innovations. Due to the ImageNet challenge, networks such as AlexNet, GoogLeNet and ResNet have proven that it is possible to have ever deeper structures capable of training on enormous datasets of images and learning their most complex features autonomously.

Analyzing those models, we realized that modern CNNs have some shortcomings associated with their increasing depth. Some of the major problems found are the amount of labeled samples and computational power required by these networks to successfully learn the data. Additionally to these problems, recent studies have showed a certain degradation of the generalization ability of such networks, caused by the excessive increase of their depth.

As a solution to these shortcomings, we implemented a classification architecture with the goal of performing image recognition tasks using approaches that differ from the stat-of-the-art deep CNNs. The main idea behind our solution was to extract the image's features and to have independent Neural Networks learning different feature types (modalities). The implemented classification architecture consists of three sequential modules: Feature Extraction, Feature Learning and Multimodal Fusion.

On the Feature Extraction module, we used computer vision techniques to extract the features and organize them in feature vectors, which would represent each modality. The color information was extracted using color histograms, which were improved with the addition of a background removing mask. For the Texture Modality, we used the Sobel operator to

compute the gradient of the images. From the gradient, we were able to create Histograms of Oriented Gradients. At last, for the shape information, we created our own approach, which consists of a weighted combination of: the silhouette of the object present in the image; the contours obtained with the Canny Edge Detector; and the corners extracted with both the Harris and Shi-Tomasi Corner Detectors.

Having extracted the features and created the feature vectors for each modality, the next step is learning the modalities on the Feature Learning module. Here, we created three Neural Networks for each modality and trained them using their respective feature vectors. At the end, we decided on a model to represent each modality, based on their performance with the validation set. We used the loss and accuracy metrics to evaluate the models performance.

With a trained model chosen to represent the Color, Texture and Shape modalities, we processed the test set with the Feature Extraction and Feature Learning modules, in order to obtain the Softmax probabilities for each test sample. These probability vectors were used as mass functions on the Dempster-Shafer's theory of evidence, on the Multimodal Fusion module. Resorting to the Dempster-Shafer's combination rule, we combined the Softmax probabilities produce by each Modality model and obtained the final labels for the test set.

To support the implementation of the proposed solution and experiment with the models from the Feature Learning module, we chose the Fruits 360 dataset. This dataset has a total of 90,380 images, assign to one of 113 labels. We used 75% of the dataset to train and validate the modules and 25% to test the Multimodal Fusion module.

The implemented solution yielded good results, having achieved an accuracy of 94.80% for the fusion of all three modalities and 97.40% for the fusion of the Color and Texture modalities. Our Multimodal classifier performed better than both the baseline CNNs trained over the same dataset, in the same circumstances: 10 epochs with a batch size 32.

We compared our results with the model used by the creators of the Fruits 360 dataset. Their best experiment achieved an accuracy 1.26% higher than our top result. However, because our solution uses shallow NNs trained over 10 epochs with samples with considerable low dimensionality, instead of a 12-layer CNN trained over 25 epochs with 100x100x3 images, we conclude that the difference of accuracy is an acceptable trade-off.

From the implementation the proposed solution and the results obtained with the experiments, the main conclusion we retain is that sometimes it makes sense to analyze the modalities independently and even combining some of them, in order to obtain similar, or even better, results than using a deep CNN. Considering as an example the Fruits 360 dataset, we were able to achieve better results than the shallow baseline CNNs, and very close results to the 12-layer CNN used by Muresan and Oltean, combining only the information of color and texture extracted from the images, through an approach that is less demanding in terms of computational power.

A. Future Work

In terms of future work for our classification architecture, the following objectives stand out:

- We would like to experiment with other feature types, such as wavelets. Considering that this feature type is different in nature from the ones we are already using, it would be interesting to analyze the results of using the time-frequency information obtained from the wavelets;
- Obtain better hardware resources in order to experiment with training over more epochs to analyze if it would improve the already good results we achieved;
- Train our Multimodal classification model with bigger datasets, such as the ImageNet dataset, which is the dataset used in the ImageNet Large Scale Visual Recognition Challenge. This dataset has 14 million images distributed over 20,000 labels.

REFERENCES

[1] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W. & Jackel, L. Backpropagation applied to handwritten zip code recognition. *Neural Computation*. **1**, 541-551 (1989)

[2] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*. **36**, 193-202 (1980)

[3] Hubel, D. & Wiesel, T. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal Of Physiology*. **160**, 106-154 (1962)

[4] Hubel, D. & Wiesel, T. Receptive fields and functional architecture of monkey striate cortex. *The Journal Of Physiology*. **195**, 215-243 (1968)

[5] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. Going deeper with convolutions. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 1-9 (2015)

[6] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 770-778 (2016)

[7] Shafer, G. A mathematical theory of evidence. (Princeton university press,1976)

[8] Mureşan, H. & Oltean, M. Fruit recognition from images using deep learning. *ArXiv Preprint ArXiv:1712.00580*. (2017)

[9] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. & Others Gradient-based learning applied to document recognition. *Proceedings Of The IEEE*. **86**, 2278-2324 (1998)

[10] Wandell, B. & Thomas, S. Foundations of vision. *Psyccritiques*. **42** (1997)

[11] Gonzalez, R., Woods, R. & Others Digital image processing. (Prentice hall Upper Saddle River, NJ,2002)

[12] Comer, M. & Delp III, E. Morphological operations for color image processing. *Journal Of Electronic Imaging*. **8**, 279-289 (1999)

[13] Dalal, N. & Triggs, B. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference On Computer Vision And Pattern Recognition (CVPR'05)*. **1** pp. 886-893 (2005)

[14] Sobel, I. & Feldman, G. A 3x3 isotropic gradient operator for image processing. *A Talk At The Stanford Artificial Project In*. pp. 271-272 (1968)

[15] Canny, J. A computational approach to edge detection. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **8**, 679-698 (1986)

[16] Harris, C., Stephens, M. & Others A combined corner and edge detector. *Alvey Vision Conference*. pp. 10-5244 (1988)

[17] Shi, J. & Others Good features to track. *1994 Proceedings Of IEEE Conference On Computer Vision And Pattern Recognition*. pp. 593-600 (1994)

[18] Schocken, S. & Hummel, R. On the use of the Dempster Shafer model in information indexing and retrieval applications. *International Journal Of Man-Machine Studies*. **39**, 843-879 (1993)

[19] Krizhevsky, A., Sutskever, I. & Hinton, G. Imagenet classification with deep convolutional neural networks. *Advances In Neural Information Processing Systems*. pp. 1097-1105 (2012)