Development of android-based interface to determine color additives in food embedded with convolution neural networks technique

W Pribadi¹, R E Masithoh¹, A P Nugroho¹ and Radi¹

¹Department of Agricultural and Biosystems Engineering, Faculty of Agricultural Technology, Universitas Gadjah Mada, Indonesia

E-mail: evi.ugm@gmail.com

Abstract. Recent advanced technology enables Android smartphone suitable for quality evaluation of food. In this research, image processing technique was used to detect food color additives. In this research, a smartphone application was developed to determine the availability of color additives in food products. Local food namely geplak was made by adding food grade (i.e. tartrazine and erythrosine) and non-food grade (Rhodamin B and Methanyl Yellow) additives in three concentrations. A mobile phone captured geplak images resulting 1200 images which were divided into 1000 images for training and 200 images for validation. Image data was processed with the python programming language of tensorflow function. The output of python in nominal weight was then trained and tested by using a convolutional neural networks (CNN) method. The weights were then processed by Android Studio version 3.2.1 using .java as backend from CNN and .xml as an application layout. Validation result showed that the program successfully determined class of food additive in high degree accuracy of 98 %.

1. Introduction

Artificial colorants are used to improve original color of products usually to give more attractive appearance. Artificial colorants provide uniformity and applicability compared to natural colorants [1]. Several artificial colorants, such as tartrazine, erythrosine, amaranth, or acid are permitted but pararosaniline (PA), auramine O (AO), methanyl yellow (MY), and rhodamine B (RB) are not permitted since they are carcinogenic and toxic to humans and animals.

Several cases of fraud in additives were found on food circulated in the community. Research conducted by the Ministry of Health in 2013 and 2014 showed that the highest agents in snacks for school children contained bacteria, cyclamate use, and textile dyes. Study in India found around 11% of food sample contained non-permitted color [2]. Non-permitted colorants are prohibited for food application regardless their concentration, but for permitted colorants although they are safe but their use need to consider the acceptable daily intake (ADI).

Detection of non-food grade colorant usually done by using special kits, chromatography [1], HPLC [3], Thin Layer Chromatography (TLC) method, or mass spectroscopy [4,5]. Rhodamine B (RB) is the oldest and most widely used synthetic dye and has been designated as an illegal additive in food in Europe and China. Initially RB was used for dyes in the textile and plastic industries, chemical analysis, and biological studies. The method of determining the content of RB in chili and its derivative products was developed by Qi et al. (2014) with HPLC combined with fluorescence detection (FLD). Those
methods require sample preparation and certain laboratory equipment which are timely and labor intensive and expensive. Apart from methods mentioned previously, non-destructive methods were also applied for determination of color pigments in food, such as UV-VIS-NIR and Raman spectroscopy [6,7]. Those methods were rapid and timeless but currently the instruments are still expensive.

As colorants are easy to visualize, color additives can be detected from images through color machine vision system. The system will capture, process, and analyze images without contact or destruct samples [8]. With the development of digital camera and computer technology, it is now becoming easier to analyze and quantify colors comprehensively compared to conventional color instruments [9,10]. The current development of smartphones supported by excellent operating system, methods for detection of color in food can be rapidly executed [11]. In this research, an image processing program based on Python programming language and Android Studio was developed. The program provided color values in RGB and Lab format. Color information was used to determine color additives of food products using convolution neural network technique.

Convolutional Neural Networks (CNN) is one of deep learning technique commonly used in visual or speech recognition. CNN is a development of multilayer perceptron (MLP) that is designed to process two-dimensional data in the form of images. Multi-layer artificial neural network architectures are now developing started by LeNet-5, ZFNet, VGGNet, GoogleNet, and ResNet [12]. All those constructions basically consist of three types of layers, namely convolutional, pooling, and fully-connected layers [13].

The convolutional layer learns feature of inputs consisted of several convolution kernels used to compute different feature maps. Each neuron in a feature map is linked to a region of neighborhood neurons which are referred to as the neuron’s receptive field in the previous layer. By convolving the input with a learned kernel and followed by using an element-wise nonlinear activation function on the convolved results, new feature map is obtained. Each feature map is generated from the kernel which is shared by all spatial locations on the input. By using several different kernels, the complete feature maps are obtained. A pooling layer is used to reduce the dimensions of feature maps and network parameters. Pooling layers computation involve neighboring pixels which is called translation invariant. The final layers in CNN architecture are fully connected layers which convert the 2D feature maps into a 1D feature vector, for further feature representation.

Recently, CNN study related to food is increasing. Ciocca, et al. (2018) applied CNN-based features for food recognition and retrieval. They used different kind of architecture to obtain the best result. A CNN model was developed to detect cleanliness of online food which resulted in good model (Chen, et al., 2018). Kiwi and apple tree branch detection on site was also successfully done by using CNN [16,17]. This research aimed at developing CNN color image based features and developing Android-based application to determine class of food grade (i.e. tartrazine and erythrosine) and non-food grade (Rhodamin B and Methanyl Yellow) colorants.

2. Materials and Methods

2.1. Materials

Samples used in this research were local food namely Geplak. Geplak was made of rice flour of 100 gr, shredded coconut of 134 gr, sugar of 67 gr, water of 60 ml, and salt of ¼ teaspoon. Food-grade colorants were added i.e. Eritrosin CI 45430 for giving red color with concentration of 280, 300, 320 mg/kg and Tartrazin CI 19140 for yellow with concentration of 50, 70, 90 mg/kg. Non-food grade colorants were also added i.e. Rhodamin B making red color and Methanyl Yellow making yellow color with concentration of 280, 300, 320 mg/kg and 50, 70, 90 mg/kg, respectively. Rice flour was mashed and roasted until dry then shredded coconut were added. Sugar was boiled in water and salt until well mixed. The solution was then added to the mix of flour and shredded coconut and finally colorant was added. The dough was then air-dried and geplak was produced. Picture of Geplak on various colorant were shown on Figure 1.a. to 1.d.
2.2. Image capture
Images of geplak were taken using a CCD camera from the XMIOMI REDMI 3 smartphone with a 13 megapixel rear camera specification with Geo Tag camera software features and video resolution of 1080p / 30fps. The image was in .jpeg format. The total image captured were 1200 images which comprises of 300 images of each concentration of colorants. There were 4 types of colorants, i.e. Erythrosine, Tartrazin, Rhodamin B, as well as Methanyl Yellow, and each colorant had 3 concentrations. Of 1200 images, 1000 data 200 images were used as training and testing in CNN method.

2.3. Development of Convolutional Neural Networks (CNN) Architecture.
There was two stages in CNN architecture development namely feature learning and classification. Flow chart for learning and classification process was shown in Figure 2. Data training will follow the Inception-v3 model, which was a model of deep learning made by GoogleNet for machine learning and has been trained using the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 dataset. There were 4 stages in training, namely: data input; image crop; bottleneck value calculation; and CNN architecture design. Data used as input was image of geplak previously taken using a smartphone. The image data was processed using Python software version 3.7.3 (Python Software Foundation, https://www.python.org/). The data analysis method used was CNN method aimed to classify geplak image containing colorants, i.e. Eritrosin cl 45430, Tartrazin cl 19140, Rhodamin B, and Methanyl Yellow. The software used library address (Directory library) in tensorflow function which was installed via the command prompt (CMD) on Windows 8.1. After inputting the image data, cropping of image was done to increase size of 224x224x3 pixel. Each image bottleneck value was then calculated to be used for the classification process. When the bottleneck value was completed, training process for final layer by using the CNN started. The output of the bottleneck process was the weight that contained the nominal layer of the image as input for CNN.
Training was conducted using total training steps of 500 with the default from GoogleNet original conditions having a maximum of 4000 steps. In the training process, the input image was 64x64x3. Value 64x64 and 3 represented number of pixels in the image and an image having 3 channels, i.e. Red Green Blue (RGB). Images were processed through convolution and pooling processes at the feature learning stage. This study applied 2 convolution layers. Each convolution and pooling layer have different filter and kernel sizes. Figure 3 showed architecture of CNN model. The image input on this model was 64x64 number of pixels. Here terms of batch and kernel were found. Batch size was the number of samples propagated into the neural network architecture. At the convolutional layer, the first number of filters was 64 with kernel size of 3x3. The kernel was a matrix to calculate and detect a pattern used during the convolution process. Pooling was process of reducing the dimensions of feature maps. As shown in Figure 4, the image had convolutional layers with 32 filters with 3x3 kernel size and pooling layers with 32 filters with a 2x2 kernel size. The feature map from the pooling layer was then converted into the direction vector as the output of the fully connected layer. Output of the CNN training process was model weights with the ability to classify images based on color. The resulting model will be stored in graph (file.pb) and label (file.txt).
2.4. Development of Computer Vision System (CVS) with Android Studio Program

CVS developed was Android-based system intended to detect colorant added in particular food in this study was geplak. The CVS method was an initial design and development of Android-based food detection applications. The Android-based application used Android Studio version 3.2.1 (Android Studio, https://developer.android.com/studio). Data was processed using a Python language which processed and analyzed image data with tensorflow using the CNN method. The output of CNN was weights of the model which were then used as input for the CVS development using Android Studio 3.2.1 in .java language. In Android Studio there was a .xml programming language as an output display. The final user interface of Android Studio was installed in an Android smart-phone. At this moment, only Android in Marshmallow and above which can use this application.

2.5. Accuracy of the Program

The developed program was ran and tested using SAMSUNG DUOS C7 PRO smartphones. A total of 200 test data from samples stored in the application program were tested to measure the accuracy of the program. Accuracy was measured from the percentage of ratio of true positive to total number of testing entries.

3. Results and Discussion

3.1. Convolutional Neural Networks (CNN) Development

As shown in Figure 4, the input image size of 64x64x3 was processed using the CNN architecture which had 5 layers, i.e. first convolution layer, first pooling layer, second convolutional layer, second pooling layer, and fully connected layer. The first convolution layer process (Figure 4.a) used a 3x3 kernel and 32 filters. The convolution layer process was the process of combining different matrices to produce a new matrix value. The result of the first convolution layer was 64x64, the same as the input, because it used a padding value of 0 (matrix layer). Pooling layer was a reduction size of the matrix or spatial input which basically consisted of a filter of a certain size. The pooling process was used to get a new matrix value. Figure 4.b showed of the first pooling layer which produced a new 32x32 matrix using a 2x2 kernel pooling that was applied in two steps and operated on each slice of the input. The second convolution layer process (Figure 4.c) was the continuation of the first pooling result. The image matrix input was 32x32 having number of filters of 64 by using a 3x3 kernel. The second pooling process was similar to the first pooling process, but there was a difference in the final value of the matrix. The resulting output had an image size of 16x16, as shown in Figure 4.d. The last layer was the fully
connected layer (Figure 3). The flatten or fully connected layer converted the pooling layer output into a direction vector.

![First Convolution Layer Process](image1)

![First Pooling Layer Process](image2)

![Second Convolution Layer Process](image3)

![Second Pooling Layer Process](image4)

**Figure 4.** Illustration of first convolution, first pooling, second convolution, and second pooling layers with input of 64x64x3 images, filter of 32 kernel of 3x3, and output of 64x64x3 images (a), input of 64x64x3 images, filter of 32 kernel of 2x2, and output of 32x32x3 images (b), input of 32x32x3 images, filter of 64 kernel of 3x3, and output of 32x32x3 images (c), as well as input of 32x32x3 images, filter of 32 kernel of 2x32 and output of 16x16x3 images (d).

**Table 1. CNN Model Results**

| Layer                      | Kernel Size | Parameters |
|----------------------------|-------------|------------|
| Input Images               | -           | 12288      |
| **First Convolutional Layer** | 32x64x3     | 12288      |
| **First Pooling Layer**    | 32x32x3     | 3072       |
| **Second Convolutional Layer** | 64x32x3     | 3072       |
| **Second Pooling Layer**   | 32x16x3     | 768        |
| **Fully-Connected Layer**  | - 15360     | 15360      |
| **Output**                 | - 3         | 3          |
| **Total**                  |             | 47107      |

Based on the description of the explanation of the network architecture previously, the architecture was used for the training process. Total parameters formed from the model were 47107 neurons (Table 1). There were 8 main parameters used in the image training process, namely Training Steps, Learning
Rate, Testing Percentage, Validation Percentage, Eval Step Interval, Train Batch Size, Batch Size Test, and Validation Batch Size. Training Steps were the number of stages in the image training process. Learning Rate was the level of learning in the image training process. Percentage testing was the percentage of images for testing. Validation Percentage was the percentage of images for validation. The Eval Step Interval was the interval of the image training stage. Train Batch Size was a measure of the number of images that will be trained at one time. Validation Batch Size was a measure of the number of images that are validated at one time. The parameter values in the image training process can be seen in Table 2.

| Parameter             | Parameter Value |
|-----------------------|-----------------|
| Training Steps        | 500 steps       |
| Learning Rate         | 0.01            |
| Testing Percentage    | 10 %            |
| Validation Percentage | 10 %            |
| Eval Step Interval    | 10              |
| Train Batch Size      | 100             |
| Test Batch Size       | -1              |
| Validation Batch Size | 100             |

3.2. *VisioApps Development*

The developed program can be applied on any types and model of smartphones as long as they are Android based system. The program appearance after installation on the (Figure 5) showing several menu such as buttons for downloading previously uploaded images or for capturing images using embedded camera. After image was uploaded or captured, it was processed resulting in RGB and Lab color information, but only Lab information were used as input for CNN structure.

![Figure 5. User interface on a screen](image)
Figure 6 shown detection results for geplak contained Tatrazin and Eritrocyn which were categorized as safe colorant, and detection results for geplak contained Methanyl yellow and Rhodamin B categorized as unsafe colorant. L value obtained from the developed program were similar for all images either using safe or unsafe colorants, but geplak added with Eritrocyn showed much lower L value. Geplak contained Eritrocyn had lower a and b values compared to other colorants meaning that it was less red and less yellow. Those values resulted from several concentrations used as explained in Methodology.

![Figure 6. Detection results for geplak contained Methanyl yellow (a) and Rhodamin B (b) as unsafe colorant, and Tatrazin (c) and Eritrocyn (d) as safe colorant](image)

### 3.3. Accuracy of the program
By using 50 samples each of previously known colorants, the VisioApps program could successfully detected 50 correct out of 50 samples geplak contained eritrocyn, 48 correct out of 50 samples geplak contained tatrazin, 49 correct out of 50 samples geplak contained Rhodamine B, and 49 correct out of 50 samples geplak contained Methanyl Yellow, as shown in Table 3. In general, the developed program had the accuracy of 98%.

| Actual Class | Predicted Class | Eritrosin | Tatrazin | Rhodamine B | Methanyl Yellow |
|--------------|-----------------|-----------|----------|-------------|-----------------|
| Eritrosin    |                 | 50        | 0        | 0           | 0               |
| Tatrazin     |                 | 0         | 48       | 1           | 1               |
| Rhodamine B  |                 | 1         | 0        | 49          | 0               |
| Methanyl Yellow |             | 0         | 1        | 0           | 49              |

### 4. Conclusion
In this research, convolution neural network was used to classify geplak into four class regarding colorant used in the product, i.e. eritrocyn, tatrazin, Rhodamine B, and Methanyl Yellow. The developed program could also be used to determine color value of object. The accuracy of the program using image stored in the smartphone was 98%.
Acknowledgement
This research was funded by Ministry of Research, Technology, and Higher Education of the Republic of Indonesia in 2018, and Universitas Gadjah Mada under Recognisi Tugas Akhir program 2019. Great appreciation is delivered to Ali Topan and Nafisatul Maghfiroh for their support and assistance in finishing this work.

References
[1] Tatebe C, Zhong X, Ohtsuki T, Kubota H, Sato K and Akiyama H 2014 A simple and rapid chromatographic method to determine unauthorized basic colorants (rhodamine B, auramine O, and pararosaniline) in processed foods Food Sci Nutr. 2(5) 547-56
[2] Purba MK, Agrawal N and Shukla SK 2015 Detection of Non-Permitted Food Colors in Edibles. J Forensic Res. 4(3)
[3] Qi P, Lin Z, Li J, Wang C, Meng W, Hong H and Zhang X 2014 Development of a rapid, simple and sensitive HPLC-FLD method for determination of rhodamine B in chili-containing products Food Chem. 164 98–103
[4] Bento WDAS, Lima B and Paim APS 2015 Simultaneous determination of synthetic colorants in yogurt by HPLC Food Chem. 183 154–60
[5] Mathiyalagan S, Mandal BK and Ling YC 2019 Determination of synthetic and natural colorants in selected green colored foodstuffs through reverse phase-high performance liquid chromatography Food Chem. 278 381–3
[6] Gukowsky JC, Xie T, Gao S, Qu Y and He L 2018 Rapid identification of artificial and natural food colorants with surface enhanced Raman spectroscopy Food Control 92 267–75
[7] Martelo-Vidal MJ and Vazque M 2014 Determination of polyphenolic compounds of red wines by UV–VIS–NIRspectroscopy and chemometrics tools Food Chem. 158 28–34
[8] Luzuriaga DA and Balaban MO 2002 Color Machine Vision System: An Alternative for Color Measurement Proceedings of the World Congress of Computers in Agriculture and Natural Resources
[9] Mendoza F, Dejmek P and Aguilera JM 2006 Calibrated color measurements of agricultural foods using image analysis Postharvest Biol Technol.41(3) 285–95
[10] Vidal M, Garcia-Arrona R, Bordagaray A, Ostra M and Albizu G 2018 Simultaneous determination of color additives tartrazine and allura red in food products by digital image analysis Talanta 184 58–64
[11] Masithoh RE, Achmad B and Zharif L 2018 Development of “Smart Eye” – Smartphone Application – To Determine Image Color and Texture of Tomatoes Proceeding 2nd Int Conf Trop Agric. 53–60.
[12] Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J and Chen T 2018 Recent advances in convolutional neural networks Pattern Recognit. 77 354–77
[13] Guo Y, Liu Y, Oerlemans A, Lao S, Wu S and Lew MS 2016 Deep learning for visual understanding: A review Neurocomputing. 187 27–48
[14] Ciocca G, Napoletano P and Schettini R 2018 CNN-based features for retrieval and classification of food images Comput Vis Image Underst. 176–177 70–7
[15] Chen H, Xu J, Xiao G, Wu Q and Zhang S 2018 Fast auto-clean CNN model for online prediction of food materials J Parallel Distrib Comput. 117 218–27
[16] Fu L, Feng Y, Majeed Y, Zhang X, Zhang J, Karkee M and Zhang Q 2018 Kiwifruit detection in field images using Faster R-CNN with ZFNet IFAC-PapersOnLine. 51(17) 45–50
[17] Zhang J, He L, Karkee M, Zhang Q, Zhang X and Gao Z 2018 Branch detection for apple trees trained in fruiting wall architecture using depth features and Regions-Convolutional Neural Network (R-CNN) Comput Electron Agric. 155(August) 386–93