DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

A DEEP LEARNING MODEL FOR MARBLE QUALITY CLASSIFICATION

by
İdris KARAALİ

March, 2021
İZMİR

A DEEP LEARNING MODEL FOR MARBLE QUALITY CLASSIFICATION

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Science

by
İdris KARAALİ

March, 2021
İZMİR

M.SC THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**A DEEP LEARNING MODEL FOR MARBLE QUALITY CLASSIFICATION**" completed by **İDRİS KARAALİ** under the supervision of **ASST. PROF. DR. METE EMİNAĞAOĞLU** and we certify that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

……

Asst. Prof. Dr. Mete EMİNAĞAOĞLU

Supervisor

……	……
Assoc. Prof. Dr. Korhan KARABULUT	Asst. Prof. Dr. Ayşe Övgü KINAY
(Jury Member)	(Jury Member)

Prof. Dr. Özgür ÖZÇELİK

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

The authors would like to thank the “Haz Marble - Marble, Natural Stone, Fixing System Industry and Trade Inc.” for supplying the marble slab images and valuable managerial feedback and comments throughout this study.

İdris KARAALİ

A DEEP LEARNING MODEL FOR MARBLE QUALITY CLASSIFICATION

ABSTRACT

The basic policy of marble enterprises is to establish sustainable high-quality products in a standardized manner. Identification and classification of different types of marbles is a critical task that is usually carried out by human experts. However, marble quality classification by human experts can be time-consuming, error-prone, unreliable, and subjective. Automated and computerized methods are needed to obtain more reliable, faster, and less subjective results. In this study, a deep learning model is developed in order to perform multi-classification of marble slab images with six different quality types. Some special image pre-processing operations were applied to the images for data augmentation and a special convolutional neural network (CNN) architecture was designed and implemented. It has been observed that the data augmentation approach for marble image samples has significantly improved the accuracy of the CNN model. Some outstanding results have been obtained with the proposed CNN model, which surpassed the alternative machine learning algorithms and even equalized the human experts' classification performance.

Keywords: Convolutional neural networks, image processing, data augmentation, marble quality classification, deep learning

MERMER KALİTESİ SINIFLANDIRMASI İÇİN DERİN ÖĞRENME DESTEKLİ BİR MODEL

ÖZ

Mermer işletmelerinin temel politikası, sürdürülebilir ve yüksek kaliteli ürünleri standartlaşmış bir yöntemle ortaya koymaktır. Farklı türdeki mermerlerin tanımlanması ve sınıflandırılması, genellikle bu alandaki uzman kişiler tarafından manuel olarak gerçekleştirilen önemli bir iştir. Bununla birlikte, mermer kalite sınıflandırılmasının insanlar tarafından ve manuel şekilde yapılması oldukça zaman alıcı, hatalara fazlasıyla açık, aynı zamanda da güvenilir olmayan ve öznel bir süreçtir. Bu süreci daha nesnel ve güvenilir, çok daha hızlı ve çok daha az insan müdahalesi gerektirecek şekilde otomatik hale dönüştüren bilgi teknolojilerine dayalı yaklaşımlar ve yöntemlere büyük ölçüde gereksinim vardır. Bu çalışmada, levha mermer resimlerini işleyerek altı farklı kalite tipine göre sınıflandıran bir derin öğrenme modeli geliştirilmiştir. Veri artırımı amacıyla, orijinal mermer resimlerine özgü bir görüntü ön işleme süreci gerçekleştirilmiş ve özel bir evrişimsel sinir ağı mimarisi tasarlanıp uyarlanmıştır. Mermer görselleri üzerinde bu çalışmada uygulanan özgün veri artırımı yaklaşımının, evrişimsel sinir ağı modelinin sınıflandırma başarısı ve doğruluk değerlerini çok önemli düzeyde arttırdığı gözlenmiştir. Evrişimsel sinir ağı modeli ile alternatif yapay öğrenme algoritmalarının tamamından çok daha başarılı sonuçlar elde edildiği ve mermer işlemedeki kalite kontrol uzmanlarının performanslarına yakın başarı düzeyinde sınıflandırma yapılabildiği ortaya konulmuştur.

Anahtar Kelimeler: Evrişimsel sinir ağları, görüntü işleme, veri artırımı, mermer kalitesi sınıflandırması, derin öğrenme

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

Except for hard stones and rocks, Marble and marble rocks, which are homogeneous in terms of color, texture and building stone properties, are rarely encountered on earth. Discontinuities, color differences, presence of foreign particles or elements, block shapes and various measurements, are the main factors affecting the quality of marble. Blocks of marble, it is evaluated according to the general characteristics of the marble type, internal parameters, geological characteristics of the marble quarry, processing line and usage characteristics (Karaca, 2003). Classification of marbles according to their quality today, despite advances in hardware and software technologies, it is mostly carried out with the intervention of human labor and experts.

The quality of marble is generally, it can vary depending on the density and size of the marble as well as the fossils and stains on the marble (Karaca, 2003). In a literature study, various classical approaches or methods for the classification of marble quality have been presented (Yavuz & Koca, 2003). Before the photographs of marble slabs are included in the quality classification systems in most of the related studies, It is known to undergo image pre-processing with different filters such as gabor, percentage, or other techniques such as extraction of chromatic properties (Bianconi et al., 2012). In the related study, a data set consisting of a total of 576 samples including 12 different quality classes consisting of pictures of colored granite tiles and 48 samples from each class was used. To minimize the distortion effect in the granite images, the images of 1,500x1,500 pixels have been reduced to 544x544 by cropping. An accuracy rate of 98.5% was obtained by using the support vector machine algorithm (Bianconi et al., 2012).

In another study in the literature on quality classification based on marble images, an industrial application has been developed for marble classification using k-means clustering algorithm. In this study, the image sizes of the marble were adjusted to be 315x310 pixels. Sampling numbers were taken as 172, 388, 411, and 187 for each distinct marble content class, respectively, and a data set consisting of a total of 1,158

samples were used (Selver, 2011). Textural characteristics used in measuring mean, variance, energy, similarity, entropy, comparison, and homogeneity were used in the marble classification - obtained using total and difference histograms (SDH) (Unser, 1986) and even (Martinez-Alajarin, Luis-Delgado, & Tomas-Balibrea, 2005). On the other hand, wavelet transform-based properties (Martinez-Alajarin, Luis-Delgado, & Tomas-Balibrea, 2003) that provide energy decomposition of a sample were obtained and these properties were used to measure the mean, median, and variance of each decomposition step. Using K-means clustering algorithm, an accuracy of 83.60% has been obtained (Selver, 2011).

Doğan & Akay, using a visual data set containing marble photographs of four different marble qualities, obtained high levels of accuracy in their study. Using the adaboost algorithm with sum and difference histograms, they have introduced a system that automatically classifies marble slabs. A data collection that consists of a total of 1,158 samples belonging to four different quality classes was used and the accuracy rates for each different marble quality class were obtained as 96.52%, 91.79%, 94.53%, and 99.25%, respectively, with the adaboost algorithm (Doğan & Akay, 2010). In another study of the researchers, results with very high accuracy were obtained by using the hierarchical radial basis function network in the same data set (Selver, 2009).

Ferreria and Giraldi, proposed a method that uses convolutional neural networks for the classification of granite tiles. They used 28x28 pixels black and white visuals and 32x32 pixels colored visuals in the training data included in the convolutional neural network. These datasets are image manipulated derivatives of images of granite tiles, which were also used in Bianconi et al. (2015). The original granite tile data in these studies are, it contains a total of 1,000 colored images of 1,500x1,500 pixels, belonging to 25 different classes and including 40 granite photographs in each class. The first 100 samples of images were obtained by scanning with an image processing system. The remaining 900 samples, it was obtained by rotating the original images between 10 and 90 degrees at nine different angles (Ferreira & Giraldi, 2017). These images, consist of a total of 2,809,000 samples, converted to black and white images

at 28x28 pixels for use in a pre-trained convolutional neural network for MNIST data (LeCun, Cortes, & Burges, 2019; LeCun et al., 1989). Similarly also, with CIFAR data (Krizhevsky, Nair, & Hinton, 2019), 2,116,000 color images of 32x32 were prepared to be run on a pre-trained convolutional neural network. The 5x2 cross-validation method was used to measure the performance in classification. In the training process in the convolutional neural network, learning rate was kept constant at 0.001 and the momentum value was set as 0.9. The stochastic gradient descent method was used for backpropagation of errors in the network and updating the weights, and the decay rate was determined as 0.0005. The researchers also used similar parameters during the training of colored tiles with the pre-trained CIFAR model, however, the only difference in this process was that the rate of reduction in weights was used as 0.0001. An accuracy rate of 87.26% has been achieved in colored tile images.

In a recent study, convolutional neural networks were used to classify marble slab images. In this study, experiments were carried out on 80 different marble images using various convolutional neural network architectures and different hyper-parameters (Pençe & Çeşmeli, 2019). However, no method was used to increase data in this study. Besides, only double classification has been preferred to distinguish marble images according to their quality, in industrial applications, marbles should be separated into more than two quality levels. It was stated that the highest accuracy rate obtained in the study was around 75%.

In this study, in the marble industry, a new marble classification model has been introduced, which can make the correct classification very quickly and effectively, at least at the level of marble experts, in an automated manner, to minimize human intervention and human labor requirements. Unlike all known studies in the literature so far, the successful results of convolutional neural networks at this level in the multiple classifications of marble quality and the unique use of data augmentation techniques that increase the accuracy of convolutional neural networks in a way that prevents overfitting is a new approach that reveals the importance of this study.

CHAPTER TWO
LITERATURE REVIEW

Marble quality classification is still done with human power despite today's technologies. Marble quality can vary according to the density and size of marbles on the marble, fossils, and stains on the marble. This field can be viewed with different perspectives on the quality classification in the course of various studies. In this field study, different data sizes, different types of data and results obtained with different methods help guide this work. In the vast majority of work done in this area, the systems were fed after the pictures had been pre-processed.

2.1 K-Means Clustering for Marble Classification (SDH and Wavelet)

One of the works done in the field of marble classification is the classification of marble quality with k-means and the establishment of the industrial model of this software. In this work, the pictures are in the size of 315x310 pixels. The sample numbers for each consistency category (from 1 to 4) are 172, 388, 411, and 187, respectively, consisting of a total data set of 1158 samples (Selver et al., 2011). The data is normalized to (0,1) and input to the system as input. In the literature (Martinez-Alajarin, Luis-Delgado, & Tomas-Balibrea, 2005) for marble (limestone) classification, textural characteristics extracted using total and difference histograms (SDH) (Unser, 1986) and used for computing characteristics of mean, variance, energy, similarity, entropy, contrast, and homogeneity were used. Wavelet transform-based characteristics, on the other hand (Martinez-Alajarin, Luis-Delgado, & Tomas-Balibrea, 2003), seek to obtain a sample's energy level and verify whether it is within a frequency band and then determine the mean, median, and variance of each decomposition level (Selver et al., 2011). With the use of k-means clustering + SDH + Wavelet layers, numerous two- and three-level hierarchical clustering techniques have achieved 83.60% percent precision.

2.2 Adaboost for Marble Classification

Doğan and Akay report on an experiment focused on four distinct marble grades with good classification accuracy. The authors recently proposed a method based on total and difference histograms and adaboost, for automated classification of marble (limestone) slabs. The sample numbers for each consistency category (from 1 to 4) are 172, 388, 411, and 187, respectively, consisting of a total data set of 1158 samples (Selver et al., 2011; Doğan & Akay, 2010). By using the SDH process, the textural characteristics are removed.

The three different versions (real adaboost, gentle adaboost, and modest adaboost) of the adaboost algorithm is used in their simulations. The results show that adaboost better than other methods that were reported previously (Selver et al., 2009).

2.3 Multilayer Perceptron for Marble Classification

"The numbers of samples for each quality group (from 1 to 4) are 172, 388, 411, and 187, respectively, comprising a data set of 1158 samples in total" (Selver et al., 2011; Selver et al., 2009; Doğan & Akay, 2010). RGB color space has been chosen and used to make its computing simplicity efficient. "MLP shows the best performance with correct classification rates of 96.52%, 91.79%, 94.53%, and 99.25% for Groups 1–4, respectively".

2.4 Pre-Trained Models for Marble Classification

The computer vision research group has recently gained significant interest from convolutional neural networks (Ferreira & Giraldi, 2017). Ferreria and Giraldi (2017) propose a methodology for granite tiles classification through the application of CNNs. They use 28x28 pixels grayscale images and 32x32 pixels colored images in their CNNs. They used the same granite tiles dataset applied in the work of Bianconi et al. (2015). "It contains 1,000 RGB images with 1,500x1,500 pixels dimensions, subdivided into 25 classes containing 40 images per class. The first 100 images were

acquired naturally by a specific scanner. The other 900 were created by rotating the initial 100 images in 10°, 20°, 30°, 40°, 50°, 60°, 70°, 80°, and 90° ". The dataset used in the experiments includes 2,809x1,000 = 2,809,000 small grayscale images to be used by MNIST. 32x32 pixels valid blocks, creating this way 2116 valid RGB blocks per image and a total of 2,116x1,000 = 2,116.000 tiny color images to be applied CIFAR. They considered using 5x2 cross-validation. Using batches of images containing 100 images, the MNIST based networks were educated, with a learning rate set at 0.001. With momentum equal to 0.9 and weighting decay equal to 0.0005 without dropout, they used the stochastic gradient descent. On the other hand, they used the same parameters for The CIFAR. The weight decay is 0.0001 without dropout. They have reached to 87.26% accuracy with CIFAR (Ferreira & Giraldi, 2017)

2.5 Support Vector Classifier for Marble Classification

For of consistency group (from 1 to 12), the number of samples is 48, containing a data set of 576 samples in total (Ferreira & Giraldi, 2017). Images cropped to 544x544 pixels to discard distortion. They have used 50% of the dataset for validation, 50% of the dataset for training (1/2). At the conclusion of the course, the tests were replicated using three distinct amounts between the prepared samples and the overall number of samples, numbered 1/2, 1/4 and 1/8. These correspond, respectively, to 32, 16 and 8 training samples per class. They have reached 98.5% accuracy using support vector machine classifier.

2.6 Deep Learning in Marble Slabs Classification

One of the works done in the field of marble classification is the classification of marble quality with a convolutional neural network which is a well-known image classification deep learning algorithm. Deep learning is a subfield of algorithm-related machine learning inspired by the brain structure and function called artificial neural networks. Deep learning is a first-class algorithm with a scalable structure that gets stronger as the number of data increases as shown in Figure 2.1 (Andrew, 2015). Another frequently cited advantage of deep learning models, in addition to scalability,

is their ability to perform automated feature extraction from raw data, also called feature learning (Brownlee, 2016). Methods in deep learning strive to learn feature hierarchies with features created by the composition of lower-level features from higher levels of the hierarchy. Learning features automatically at several abstraction levels helps a machine to learn complex functions that map the input to the output directly from code, without relying solely on human-made features (Bengio, 2019).

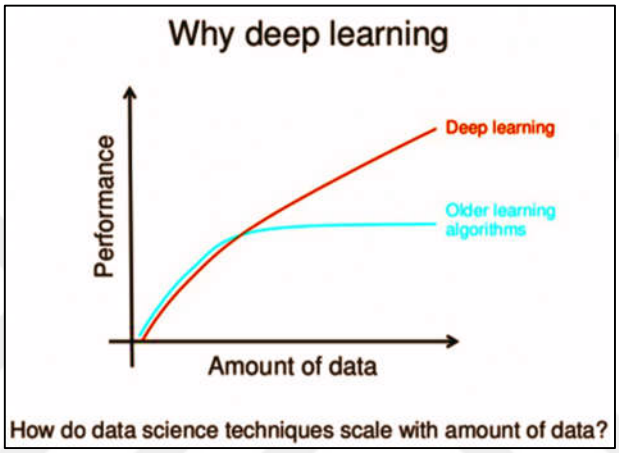


Figure 2.1 What is deep learning? (Brownlee, 2016)

Pençe & Çeşmeli used 80 marble samples to suggest a new class definition for travertine slabs taken for Başarırlar Marble Company in 45x45 cm dimensions (Şişeci & Cetişli, 2012). In this study, they are worked with 9 classes such as 5 first quality class, 3-second quality, and non-homogenous ones. Stochastic gradient descent with momentum (sgdm), RMSprop and adam methods used for optimize model accuracy. After 300 iterations, they have reached to 75% accuracy with adam optimizer.

2.7 Convolutional Neural Network

A convolutional neural network (CNN or ConvNet) is a class of deep neural networks used most frequently to interpret visual images in deep learning (Zhang et al., 1990). Convolutional neural networks are biologically-inspired variants of multilayer perceptrons intended to mimic visual cortex behavior (Hanbay, 2020; Convolutional neural network, 2019).

2.7.1 Convolution Layer

The first layer of a convolutional neural network model is the convolution layer (Chakraverty, 2018). The convolution layer that gives the network its name is fundamental to the convolutional neural network. A process called a "convolution" is performed by this layer (Brownlee, 2019).

The breakthrough of convolutional neural networks is the capacity of automatically learn a large number of filters in parallel unique to a training dataset under the constraints of a particular predictive modeling task, such as image classification. The effect is a highly specialized function that can be seen everywhere on input images (Brownlee, 2019).

Convolution is a linear operation in the sense of a convolution neural network that involves the multiplication of a set of weights with the input, much like a traditional neural network. The multiplication is carried out between an array of input data and a two-dimensional array of weights, called a filter or a kernel, provided that the technique was constructed for two-dimensional input (Brownlee, 2019).

The filter is smaller than the input data and the type of multiplication applied between an input filter-sized patch and the filter is a dot product. A dot product is an element-wise multiplication, often resulting in a single value, between the input filter-sized patch and the filter that is then summed up as shown in Figure 2.2. The process

is often referred to as a scalar product because it results in a single value (Brownlee, 2019).

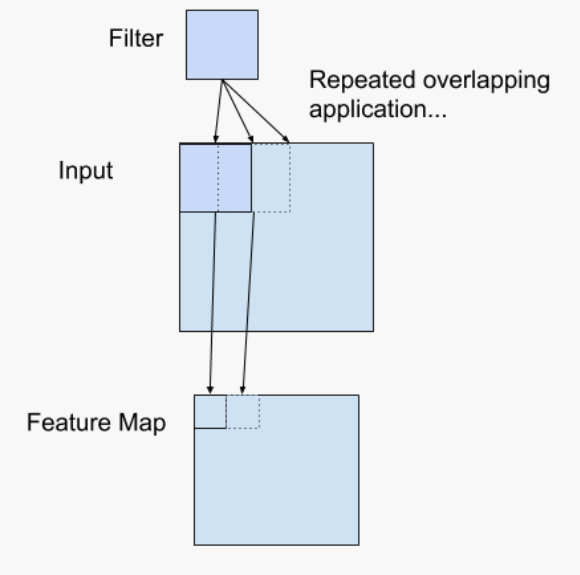


Figure 2.2 Filter Example Applied to a two-dimensional input to construct a function map

2.7.2 Activation Functions

We used rectifier linear unit (ReLU) and softmax activation functions in this analysis. In 2000, Hahnloser et al. first applied the activation function of ReLU to a complex network with strong biological motives and mathematical justifications ReLU (Rectifier (neural networks), n.d.) activation functions as mentioned below eq. (2.1), are defined where x is the neuron input.

$$f(x) = \max(0, x) \tag{2.1}$$

In the model's final sheet, the softmax function is used to estimate the groups. The output values are pleasant between the range[0,1] since in our neural network model we can avoid binary classification and accommodate as many classes or dimensions (Softmax function, n.d.). The function of softmax is defined below eq. (2.2) and works as shown in Figure 2.3.

(2.2)

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

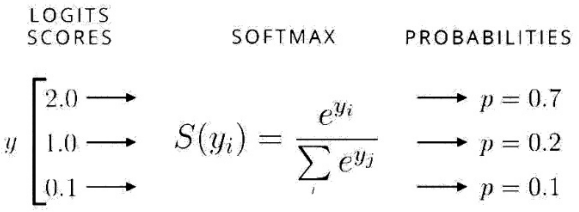


Figure 2.3 Softmax function works

2.7.3 First-Order Optimization Algorithms

First-order optimization algorithms specifically require selecting the path to travel in the search space by using the first derivative (gradient). The methods include first measuring the feature gradient, and using a phase size to follow the gradient in the opposite direction (e.g. downhill to the lowest for minimization problems) (also called the learning rate) (Brownlee, 2020).

First-order algorithms are commonly referred to as gradient descent, with more particular names relating to slight procedural extensions, such as (Brownlee, 2020):

- Gradient Descent
- Momentum
- AdaGrad
- RMSProp
- Adam

The gradient descent algorithm also provides the template used to train artificial neural networks (deep learning) models for the common stochastic variant of the algorithm, called stochastic gradient descent (SGD) (Brownlee, 2020).

Used in the final model, RMSProp attempts to minimize oscillations, but differently than momentum. The RMS prop often removes the need, and does this automatically,

to change the learning rate. Also, for any parameter, RMSProp selects a different learning rate. Each update is performed according to the equations described below in the RMSprop as shown in Figure 2.4. This modification is performed independently for each parameter.

$$For\ each\ Parameter\ w^j$$
$$(j\ subscript\ dropped\ for\ clarity)$$
$$\nu_t = \rho\nu_{t-1} + (1-\rho) * g_t^2$$
$$\Delta\omega_t = -\frac{\eta}{\sqrt{\nu_t + \epsilon}} * g_t$$
$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

η : *Initial Learning rate*
ν_t : *Exponential Average of squares of gradients*
g_t : *Gradient at time t along* ω^j

Figure 2.4 RMSprop Function (Kathuria, 2018)

2.7.4 Fully Connected Layer

For computer vision tasks, the classic neural network architecture was found to be inefficient. The picture data includes a large neural network input. This layer takes an input volume and outputs an N-dimensional vector (whatever the output is of the convolution, or ReLU, or pooling layer preceding it), where N is the number of classes the program can pick as shown in Figure 2.5 (Desphande, 2019).

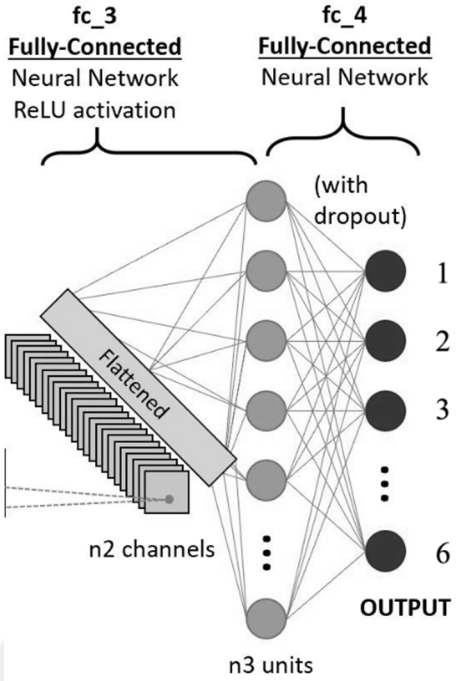


Figure 2.5 Fully Connected Layer from Proposed Model

2.7.5 Max Pooling

Maximum pooling or peak pooling is a method of pooling that in each patch of each function map determines the maximum or largest value. The findings are downsampled or pooled feature maps that highlight the most recent feature in the patch, not the feature's average appearance in an average situation of pooling. It has been shown that this performs better than average pooling for computer vision activities in practice, such as image recognition as shown in Figure 2.6.

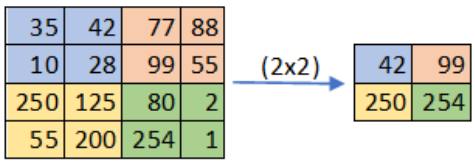


Figure 2.6 Max pooling example with 2x2 kernel

2.7.6 Dropout Rate

Dropout is a strategy where, during preparation, randomly chosen neurons are overlooked. They are spontaneous "dropped-out". This means that their contribution to the activation of downstream neurons on the forward pass is momentarily excluded and any weight changes on the backward pass are not added to the neuron as shown in Figure 2.7.

Neuron weights settle into their context within the network as a neural network knows. Neuron weights are tuned for particular functions to offer some specialization. This specialization is dependent on adjacent neurons, which, if taken too far, may lead to a shaky model too specialized for training data. This is referred to as dynamic co-adaptations, based on the background of a neuron during preparation.

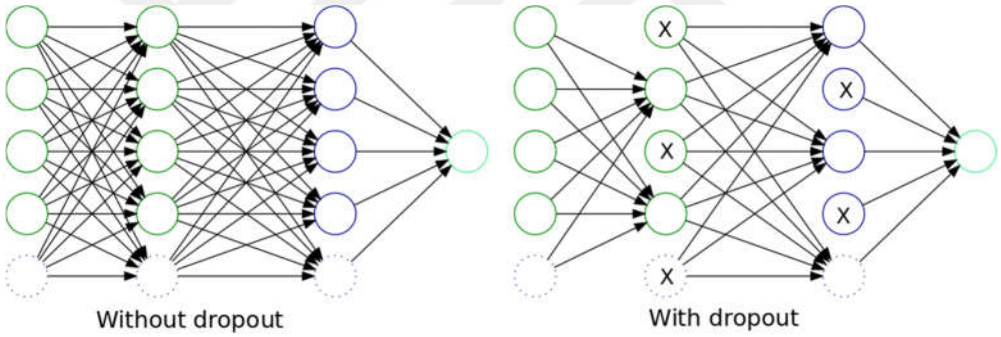


Figure 2.7 How dropout works (Bonnin, n.d.)

2.7.7 Cross Entropy

As part of the optimization algorithm, it is necessary to constantly estimate an error in the current state of the model. This involves the option of an error function, usually referred to as a loss function, that can be used to approximate the model's loss such that the weights can be changed at the next measurement to minimize the loss.

Cross-entropy measures a score that summarizes the average variance among all groups in the problem between the real and expected probability distributions, as seen in eq. (2.3). The ranking is reduced and there is a complete cross-entropy value of 0.

$$L(\theta) = -\sum_{i=1}^{k} y_i \log(\hat{y}_i) \tag{2.3}$$

2.7.8 Learning Rates

The learning rate is a hyper-parameter that controls how often our network weights are modified with respect to the loss gradient. The lower the value, the slower the downward slope we move down. Although this could be a smart strategy to ensure that we do not skip any local minima (using a low learning rate), it may also mean that we would take a long time to converge, particularly if we get stuck on a plateau area. Learning rates are usually naively set by the consumer at random. At best, to obtain an insight into what is the best value to use in setting learning rates, the consumer can exploit previous interactions (or other forms of learning material) (Zulkifli, 2018).

CHAPTER THREE
PROPOSED MODEL

The proposed model firstly was developed for Marble Companies, whose main business activity is marble manufacturer, to evaluate the marble quality classification processes.

First of all, based on the classifying method, the convolutional neural network was built as shown in Figure 3.1. For each quality class, the number of samples is 350 images, comprising a data set of 2,100 samples in total. Details about the implemented approach were explained in the Design and Implementation section.

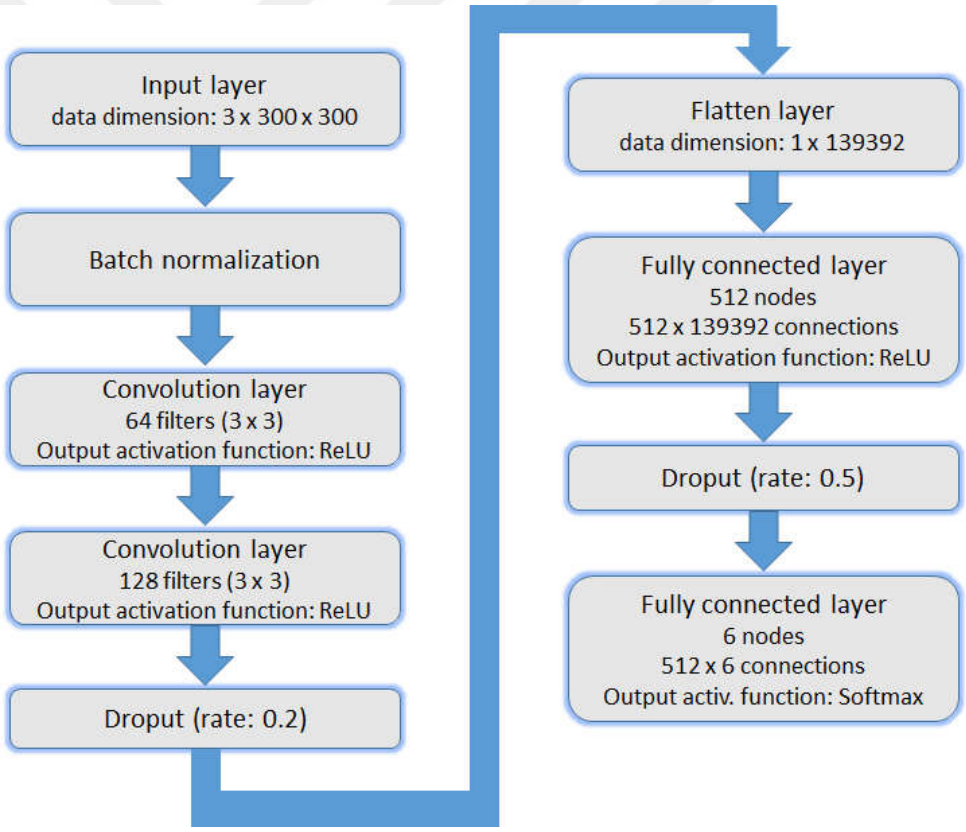


Figure 3.1 Architecture of our convolutional neural network model

3.1 Materials

The marbles consist of 90-98% calcium carbonate and the remainder of magnesium carbonate. Marbles and marble rocks, which are homogeneous in terms of color, texture and building stone properties, are rarely encountered on earth. Discontinuities, color differences, presence of foreign particles or elements, block shapes and various measurements, are the main factors affecting the quality of marble. The quality of the marble is determined by experts today. When we researched the idea of automatic classification of marble quality by machines, we discussed the following questions.

- Hardware - are fees scalable?
- Model Accuracy – is it more successful than the experts?
- Environment - is the lighting of the environment where the product is made suitable for taking pictures?
- Prediction speed - faster result than experts?

The marble slab images used in this study is the actual production data provided by "Haz Marble - Marble, Natural Stone, Fixing System Industry and Trade Inc.", which is an international company founded in Izmir, Turkey. The pictures which create the data set has been collected from the marbles produced on the production line. The dataset is compiled from pictures taken during production, quality control pictures and sales department's pictures. Some of the marbles from collected pictures showing the end product and some of them are showing the available slabs for cutting. The big slabs 4,000x4,000 pixels has been cut with python code as 300x300 pixels and added to data set is shown in the Figure 3.2 and Figure 3.3.

```
while img.width > sizewidth:
    sizeheight = counter_height * 300
    left, top, width, height = sizewidth, sizeheight, sizewidth + 300, sizeheight + 300
    while img.height > sizeheight:
        left, top, width, height = sizewidth, sizeheight, sizewidth + 300 , sizeheight + 300
        cropped = img.crop( ( left, top, width, height ) )
        cropped.save(str(counter)+'-QX-XIX.jpg')
        sizeheight = sizeheight + 300
        counter = counter + 1
    sizewidth = sizewidth + 300
    counter_height = counter_height + 1
```

Figure 3.2 Python code for crop images

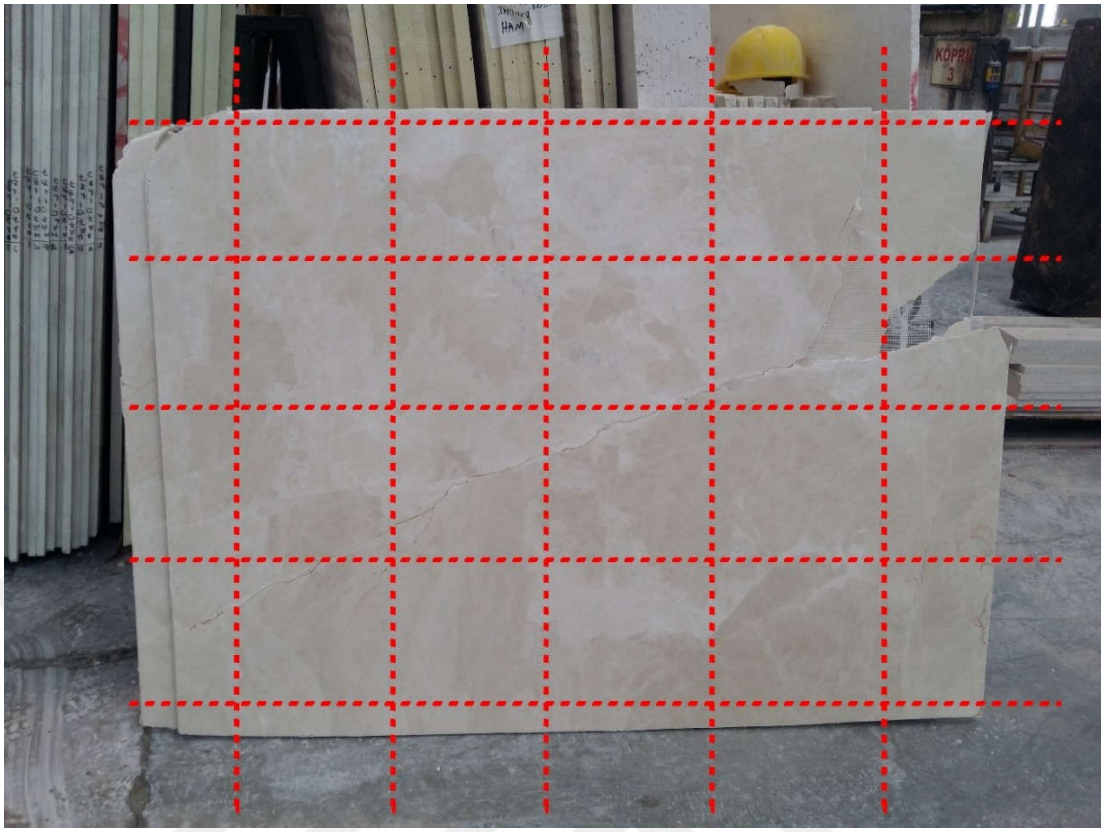


Figure 3.3 How to crop with Python code (Personal archive, 2017)

The pictures on the data set have been classified into 6 different folders. The pictures which are classified into 6 folders have been tagged by a python code along with the upper folder names is shown in Figure 3.4 (see “Q3A.1.jpg”). The pictures have been separated randomly as 1,800 train and 300 test is shown in Figure 3.5. 2,100 pieces of data needed a cluster of 2,019 images from a space of 5,000 images with a 98% confidence interval and ±2% error. At least 1 image from each class is guaranteed in the 5,000 images set.

```python
from PIL import Image
import glob, os
import sys
#name="explainer.jpg"
name= sys.argv[1]
counter = 1
for infile in glob.glob("*.jpg"):
    file, ext = os.path.splitext(infile)
    im = Image.open(infile)
    im.save(name+str(counter)+".jpg")
    counter = counter + 1
```

Figure 3.4 Rename dataset with Python code

```python
#path = "filtered/"
#folder = ['2x2Closing','2x2Dilation','2x2Erosion+Dilation','2x2Erosion','2x2Opening',
#          '4x4Closing','4x4Dilation','4x4Erosion','4x4Opening','Bilateral','Blur',
#          'GaussianBlur','GreyMedian','Median','sepFilter']
arr = range(350)
path = "data/"
rand = random.sample(arr,50)
folder =['OriginalData']
subfolder = ['Q3A','Q3B','Q3C','Q4A','Q4B','Q4C']
for fold in folder:
    for ad in subfolder:
        for x in rand:
            img = Image.open(path+fold+"train/"+ad+"/"+ad+"."+str(x+1)+".jpg")
            img.save(path+fold+"test/"+ad+"/"+ad+"."+str(x+1)+".jpg")
            os.remove(path+fold+"train/"+ad+"/"+ad+"."+str(x+1)+".jpg")
            print(str(x+1))
```

Figure 3.5 Rename dataset with Python code

For each quality class, the number of samples is 350 images, comprising a data set of 2,100 samples in total. There exist two different marble types and three different classes for each type (Q3-A, Q3-B, Q3-C, Q4-A, Q4-B, Q4-C, namely). Some sample images from each of these different quality classes are shown in Figure 3.6. and Figure 3.7. It should be remembered that the classification of marble quality is a very difficult task since most of the marble samples belonging to different grades are very similar in appearance and even a human expert can hardly differentiate them. In addition, some marble samples belong to the same quality class, but due to the colors and some patterns on the marble surfaces, they seem to be entirely different for most humans' visual perception. In other words, both the risks of false positive and false negative

categorizations of marble samples are considered to be high. These challenges and problems are also true for the dataset used in this study. As could be seen from Figure 3.8, it is very difficult both for human and computer vision to perceive and categorize these marble images in the same class.

Figure 3.6 Pictures of marble slabs from two distinct quality classes

Figure 3.7 Pictures of marble slabs from two distinct quality classes

Figure 3.8 Three marble slab images from the same class, Q4-C, namely

It is known that the higher the number of data, the better the deep learning algorithm gives. Considering this situation, filters were added to existing images to increase the number of data. In the data preparation part of the project, some filters were determined

to be applied to the images. Retinex, msrcp, and automated-retinex filters were applied to images as shown in Figure 3.9.

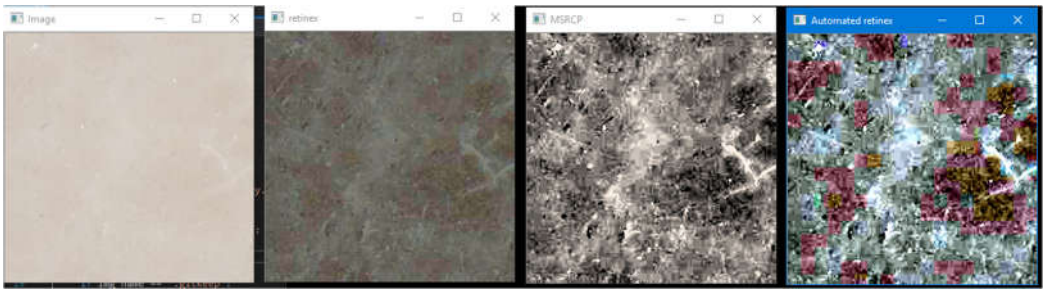


Figure 3.9 Images after filters with original image

In addition to these filters, we used some filters from the OpenCV library. One of them morphological transformations. Any basic operations based on the image shape are morphological transformations (Morphological Transformations, 2020). Normally, it is carried out on binary images. It involves two inputs, one is our original image, the second is called the kernel or structuring factor that defines the nature of the operation. Erosion and dilation are two simple morphological operators. Then it also comes into play with its variant forms, such as opening, closing, gradient, etc. (Image Filtering, 2019). We used 2x2 and 4x4 kernel for all as shown in Figure 3.11. Also used mixed transformations. For example, 2x2 and 4x4 kernel, dilation after erosion as shown in Figure 3.12. Also used bilateral filter, blur filter, gaussian blur filter, grey median filter, median filter, gabor filter, separable linear filter as shown in Figure 3.10. The parameters used in these filters are important and the values of some important parameters are given below. You can also find source code at the GitHub repository (Karaali, 2019).

```
#bilateralFilter
#bilateralFilter#bilateral = cv2.bilateralFilter(image,5,150,150)
#bilateralFilter#image = cv2.namedWeighted(image, 1.5, bilateral, -0.5, 0)
#bilateralFilter#image = cv2.bilateralFilter(image, 5, 150, 150)
#bilateralFilter#status = cv2.imwrite("Bilateraltrain/"+name+"/"+name+"."+str(x+1)+".jpg", image)

#blur
#blur# blur = cv2.blur(image,(5,5))
#blur# status = cv2.imwrite("Blurtrain/"+name+"/"+name+"."+str(x+1)+".jpg", blur)

#gaussianBlur
#gaussianBlur# gblur = cv2.GaussianBlur(image,(5,5),0)
#gaussianBlur# status = cv2.imwrite("GaussianBlurtrain/"+name+"/"+name+"."+str(x+1)+".jpg", gblur)

#boxFilter
# image=100*(image-np.mean(image))
# image[np.where(image>255)]=255
# image=cv2.boxFilter(image,-1,(30,30))
# image[np.where(image>150)]=255; image[np.where(image<=150)]=0
# image=cv2.boxFilter(image,-1,(15,15))
# image[np.where(image>0)]=255
# image = image.astype(np.float32) / 255
# status = cv2.imwrite("Boxedtrain/"+name+"/"+name+"."+str(x+1)+".jpg", image)

#erosion
# kernel = np.ones((2,2),np.uint8)
# erosion = cv2.erode(image,kernel,iterations = 1) # Finds fossils and holes
# dilation = cv2.dilate(erosion,kernel,iterations = 1)  # finds veins
# opening = cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel) # Finds fossils and holes
# closing = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel) # finds veins
# grnameient = cv2.morphologyEx(image, cv2.MORPH_GRnameIENT, kernel)
# tophat = cv2.morphologyEx(image, cv2.MORPH_TOPHAT, kernel)
#blackhat = cv2.morphologyEx(image, cv2.MORPH_BLACKHAT, kernel)
# status = cv2.imwrite("2x2Erosion+Dilation_Train/"+name+"/"+name+"."+str(x+1)+".jpg", dilation)
# status = cv2.imwrite("4x4Dilationtrain/"+name+"/"+name+"."+str(x+1)+".jpg", dilation)
# status = cv2.imwrite("4x4Openingtrain/"+name+"/"+name+"."+str(x+1)+".jpg", opening)
# status = cv2.imwrite("4x4Closingtrain/"+name+"/"+name+"."+str(x+1)+".jpg", closing)

# sepFilter2D
# kernel = cv2.getGaussianKernel( 15, 2 )
# twoD = cv2.sepFilter2D(image, -1, kernel, kernel)
# status = cv2.imwrite("sepFiltertrain/"+name+"/"+name+"."+str(x+1)+".jpg", twoD)

# Laplacian,Sobelx,Sobely test
# laplacian = cv2.Laplacian(image,cv2.CV_64F)
# sobelx = cv2.Sobel(image,cv2.CV_64F,1,0,ksize=3)
# sobely = cv2.Sobel(sobelx,cv2.CV_64F,0,1,ksize=3)
#
```

Figure 3.10 Some of image filters Python code

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 3.11 Low Pass Kernel size equation

$$G_i = \alpha * e^{-(i-(ksize-1)/2)^2/(2*sigma^2)}$$

Figure 3.12 Gaussian kernel equation

3.2 Design and Implementation

The architectural design and structure of our convolutional neural network model used in this study are shown in Figure 3.13. Experiments have been conducted with many different convolutional neural network models and very different hyper-parameters, and it has been observed that the most successful results are obtained with this architecture. Batch normalization was used in the input layer. In the next step, each visual is passed through two convolution layers and 64 and 128 filters with 15x15 kernel size and 3x3 stride are used in these layers respectively. In each convolution layer, ReLU is used as the output activation function. In the ReLU processes in the second convolution layer, a dropout of 0.20 was used. In our convolutional neural network model in this study, Unlike many known convolutional neural network architectures and models, max pooling or average pooling was not used at any stage. The reason for this is that when the pooling mechanism is added to our model, significant decreases have been observed in the accuracy rates in the classification of data belonging to two of the six classes. In the next stage of our convolutional neural network model, a flattening layer is used to transform the data into a one-dimensional index with 139,392 elements. This one-dimensional index is used as the input layer of the fully linked layer consisting of 139,392 nodes and connected to a hidden layer of 512 nodes. A dilution of 0.50 was also used in the fully connected layer and ReLU was used as the output activation function. In the last stage, 512 nodes were configured to be fully connected to six output layer nodes, and softmax was used as the activation function (as well as multi-classification function) in the output layer. Our

convolutional neural network model, each of the six nodes in the output layer is structured to represent one of six different classes of marble. During the training process of our convolutional neural network model, the size of each stack was set to 16 in stack normalization, and the training was continued for a total of 30 epochs. In order to adjust the learning rate, RMSProp one of the adaptive learning rate optimization models, was used (Buduma & Locascio, 2017; Goodfellow, Bengio, & Courville, 2016) and the initial learning rate was defined as 0.001.

Our convolutional neural network model is coded using version 3.6.7 of the Python programming language and the following software architectures and libraries: Keras 2.2.4, Tensorflow 1.13.1, PyCharm Pro. 2018.1.4 and Atom 1.34.0. The model is trained using the NVidia GTX 1050ti 4gb video card. Also, a special python code was developed to directly read images and convert them to an array. Labels are defined statically in the developed code Figure 3.13. Some parts of our code are shown in Figure 3.14 and Figure 3.15. In our experiments and tests in this study, colored marble images of 300 x 300 pixels were used at first. Although experiments were carried out with many different convolutional neural network models and very different hyper-parameters, the desired accuracy rates could not be achieved, especially if the accuracy rate of six classes reached 1.0 at the end of the training (train), excessive decreases in validation and tests were observed and overfitting (overfitting) problem has been observed. In the stack normalization phase in our convolutional neural network model, different batch sizes have been tried, and many different experiments have been performed by changing the dilution ratios. However, in all these different approaches, the problem of over-compliance has not been overcome. On the other hand, when the same marble data is tested with other alternative machine learning algorithms, an overfitting problem has been continuously observed.

```
def image_to_txt (name):
    img_matris = []
    label = []
    row = 0
    strr = ''
    strr2 = ''
    for infile in glob.glob("./"+name+"/*.jpg"):
        # print(infile)
        row = row + 1
        img = cv2.imread(infile)
        file, ext = os.path.splitext(infile)
        # print(infile)
        file = file.replace('-','.')
        labelz = file.split('\\')
        labelx = labelz[1].split('.')
        img_matris.append(img)
        str = labelx[0]
        label.append(get_label(str))
        # print(labelx[0])
        # index = 0
        # for color in range(3):
        #     for width in range(300):
        #         for height in range(300):
        #             index = index + 1
        #             strr = strr  +  str(np_img[0,width,height,color]) + ','
        # strr2 = strr2 + labelx[0]
        # file2write.write(strr+ '\n')
    labels = np.array(label)
    np_img = np.array(img_matris)
    return np_img, labels
    # file2write.close()

def get_label(argument):
    switcher = {
        "Q3A": 1,
        "Q3B": 2,
        "Q3C": 3,
        "Q4A": 4,
        "Q4B": 5,
        "Q4C": 6,
    }
    return switcher.get(argument, 1)
```

Figure 3.13 Python code for reading images and convert an array

```
batch_size = 16 #8
num_classes = 6
epochs = 30 #60
data_augmentation = True
save_dir = os.path.join(os.getcwd(), 'saved_models')

(x_train, y_train) = imgt.image_to_txt("Blursep44train")
(x_test, y_test) = imgt.image_to_txt("Blursep44test")
# print(x_train)
print(x_train.shape)
print(x_test.shape)

# Convert class vectors to binary class matrices.
y_train = keras.utils.to_categorical(y_train-1, num_classes)
y_test = keras.utils.to_categorical(y_test-1, num_classes)

model = Sequential()
model.add(Conv2D(64, (15, 15),strides=(3,3), padding='same',
                 input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(Conv2D(128, (15, 15),strides=(3,3)))
model.add(Activation('relu'))
model.add(Dropout(0.20))



model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))


# initiate RMSprop optimizer
opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)

# Let's train the model using RMSprop
model.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

Figure 3.14 Python code for the architecture and parameter settings of our convolutional neural network model

```
112
113 # Save model and weights
114 if not os.path.isdir(save_dir):
115     os.makedirs(save_dir)
116 model_path = os.path.join(save_dir, 'idris')
117 model.save(model_path)
118 print('Saved trained model at %s ' % model_path)
119
120 # Score trained model.
121 scores = model.evaluate(x_test, y_test, verbose=1)
122 print('Test loss:', scores[0])
123 print('Test accuracy:', scores[1])
```

Figure 3.15 Python code for saving the trained model and printing the test results

In order to increase the accuracy of test and validation results and to eliminate over-compliance, it was decided to try various data enhancement methods. First of all, L1 and L2 regulations were tried (Buduma & Locascio, 2017; Goodfellow, Bengio, & Courville, 2016), but it was observed that these methods did not solve the overfitting problem in our model. In addition, data augmentation, which is another strategy for eliminating excessive harmony in convolutional neural networks, and accordingly, noise injection, image rotation, color jittering, and horizontal and vertical rotation of the image. and vertical flips) and random cropping has also been tried (Goodfellow, Bengio, & Courville, 2016; Avcı, 2018). In addition, some special image processing methods and filters such as gabor filter, bilateral filter, blur filter, gaussian blur filter, box filter, morphological filters (erosion, dilation), laplacian filter, sobel filter, retinex, msrcp ve automated-retinex were also used on marble images (Karaali, 2019). Each different method is applied in all marble images, the new visuals that were formed were also included in the data set, and then the convolutional neural network was re-trained and tested. However, none of these could solve the problem of overfitting and improve accuracy rates, especially in tests. After all these unsuccessful results, it was decided to try a more original approach to eliminate the problem of overfitting, and new visuals were obtained by applying some selected image processing filters on the marble images. In the application of different image processing methods and filters on images, 82 different models were determined and tested, and each one was run separately in the convolutional neural network. Each model has been indexed according to its parameters and saved to the file. In addition, the filter selection was randomly taken from the existing filtered images folder ('2x2 Closing' ,'2x2 Dilation', '2x2 Erosion + Dilation', '2x2 Erosion', '2x2 Opening', '4x4 Closing', '4x4 Dilation',

'4x4 Erosion', '4x4 Opening', 'Bilateral', 'Blur', 'GaussianBlur', 'GreyMedian', 'Median', 'sepFilter') with python code and transferred to the models. Finally, an image processing and data enhancement approach that achieves the highest possible accuracy is introduced, overcoming the problem of overfitting for marble slab visuals and convolutional neural networks. For each marble image in the data set, new visual samples were obtained by applying blur filter and the two-dimensional linear separator filter (2D Linear Separable Filter) and these were added to the data set. Alternatively, 10-fold cross validation was used in the model instead of using the same validation data in each iteration as shown in Figure 3.16 .

```
model.add(Activation('relu'))
model.add(Conv2D(128, (15, 15),strides=(3,3)))
model.add(Activation('relu'))
model.add(Dropout(0.20))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))

# initiate RMSprop optimizer
opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)

# Let's train the model using RMSprop
model.compile(loss='categorical_crossentropy',
                        optimizer=opt,
                        metrics=['accuracy'])


kfold = KFold(n_splits=10, shuffle=True)

fold_no = 1
for train, test in kfold.split(x_train,y_train):
        print('------------------------------------------------------------------------')
        print(f'Training for fold {fold_no} ...')

        model.fit(x_train[train], y_train[train],
                                batch_size=batch_size,
                                epochs=epochs,
                                validation_data = (x_train[test], y_train[test]),
                                verbose=verbosity)

        # Score trained model.
        # Generate generalization metrics
        scores = model.evaluate(x_train[test], y_train[test], verbose=verbosity)
        print(f'Score for fold {fold_no}: {model.metrics_names[0]} of {round(scores[0],6)} -*-
              {model.metrics_names[1]} of % {round(scores[1]*100,6)}')
        acc_per_fold.append(scores[1] * 100)
        loss_per_fold.append(scores[0])

        # Increase fold number
        fold_no = fold_no + 1

scores = model.evaluate(x_test, y_test, verbose=verbosity)
print('Final-Test loss:', scores[0])
print('Final-Test accuracy:', scores[1])
```

Figure 3.16 10-fold cross-validation Python code

3.3 Results

Image blurring was achieved by convolution of the marble images with a low-pass filter (Image Filtering, 2019) consisting of a 4 x 4 matrix so that the noise and distortions in the marble paintings could be removed. Since the marble images used in this study contain many cavities (veins) and fossil remains, smoother visuals were obtained thanks to the blur filter. On the other hand, using the blur filter alone was not sufficient to distinguish the cavities and veins. In order to make the vein patterns in the marble images better distinguished, a two-dimensional linear separator filter was used (Image Filtering, 2019).

The new visuals obtained through these filtering processes applied separately to the visuals were added to the existing data set, thereby increasing the data. In this way, the number of visual examples in the data set in our study has increased to 6,300. The number of samples belonging to each of the six different marble quality classes is 1,050. 5,400 of the visuals in this data set were reserved for training and 900 were used for testing. In addition, all of the 6,300 samples were used in experiments with the 10-fold cross-validation method.

The convolutional neural network model presented in this study was analyzed in comparison with some other known artificial learning algorithms, and average accuracy rates obtained with each are presented in Table 3.1 and Table 3.2. Among the artificial learning algorithms covered in the study, especially those used for multiple classifications of visual data in the relevant literature, multilayer perceptron, k nearest neighbors, C4.5 and Naive Bayes. The multilayer perceptron is a feed-forward artificial neural network model that performs gradual descent and backpropagation, error calculation, and weight updates. It usually consists of one or several hidden layers and Sigmoid is generally preferred as the output activation function. You can flexibly adjust the learning rate, momentum, number of repetitions, and number of nodes in hidden layers (Buduma & Locascio, 2017). K-nearest neighbors (k-nearest neighbors or k-NN), it is a machine learning algorithm that learns depending on the similarity or distance between the records in the data set and defines

each record as a vector in the hyperspace (Aha, Kibler & Albert, 1991; Küçük & Balcı, 2019). Naive Bayes is an algorithm used for classification problems and based on Bayes' theorem and assumes that the attributes are completely independent of each other (Han, Pei, & Kamber, 2011; Utku & Doğru, 2017). C4.5 is a decision tree algorithm that is used for classification problems and creates the trunk, leaves, and branches of the tree using the information acquisition method (Han, Pei, & Kamber, 2011).

Table 3.1 Performance comparison of several classifier algorithms versus our CNN model using the original dataset with 2,100 images

Algorithm name and parameter values	Average accuracy rate (Training)	Average accuracy rate (Test)	Average accuracy rate (10-fold cross-validation)
Convolutional neural network model	1.000	0.714	0.701
k-NN (k nearest neighbor k = 1, Euclidean distance)	0.915	0.652	0.661
Multi-layer perceptron (one hidden layer with 64 nodes, learning rate = 0.1, momentum = 0.2, activation function: sigmoid)	0.908	0.695	0.697
Multi-layer perceptron (two hidden layers with 256 and 128 nodes, learning rate = 0.03, momentum = 0.1, activation function: sigmoid)	0.910	0.701	0.693
C4.5 Decision Tree	0.887	0.612	0.625
Naïve Bayes	0.846	0.603	0.614

The results in Table 3.1 were obtained by using the original marble images, 1,800 of these samples were separated and analyzed for training and 300 for testing, furthermore, the 2,100 samples were completely cross-validated with ten layers. As can be seen in Table 3.1, the results obtained from the original marble images, besides the fairly low accuracy rates observed in testing and validation, it is unsatisfactory due to the problem of overfitting. On the other hand, thanks to the data increase resulting from our original image processing approach, much more successful results were

obtained with a data set consisting of 6,300 images in total, and these results are shown in Table 3.2.

Table 3.2 Performance comparison of several classifier algorithms versus our CNN model using the augmented dataset with 6,300 images

Algorithm name and parameter values	Average accuracy rate (Training)	Average accuracy rate (Test)	Average accuracy rate (10-fold cross-validation)
Convolutional neural network model	1.000	0.922	0.961
k-NN (k nearest neighbor k = 1, Euclidean distance)	0.906	0.729	0.703
Multi-layer perceptron (one hidden layer with 64 nodes, learning rate = 0.1, momentum = 0.2, activation function: sigmoid)	0.893	0.748	0.757
Multi-layer perceptron (two hidden layers with 256 and 128 nodes, learning rate = 0.03, momentum = 0.1, activation function: sigmoid)	0.917	0.705	0.712
C4.5 Decision Tree	0.821	0.694	0.685
Naïve Bayes	0.735	0.655	0.661

Compared to Table 3.1, Table 3.2, it is seen that the accuracy rates of all other alternative algorithms, as well as our convolutional neural network model, have increased significantly. In addition, when Table 3.2 is examined, the convolutional neural network model, it is seen that both test results and cross-validation results provide much higher success rates than all other algorithms and the problem of overfitting is not at a high level. As can be seen in Table 3.2, the average accuracy rate obtained as a result of the test with the convolutional neural network model is 0.922, and the average accuracy rate obtained with the artificial neural network of the second-best result is only 0.748. Similarly, the average accuracy rate was observed as 0.961 as a result of ten-fold cross-validation with the convolutional neural network model, and the average accuracy rate of the multi-layer perceptron type artificial neural network, which gave the second most successful result, was only 0.757.

In the study of Bianconi et al. (2012), different from the hardly distinguishable marble images, granite images were used and a granite classification was made. The formation of granites takes more years than marbles and therefore granites contain more crystallized substances. Distortions in the form of clumps are generally observed in granites, but veins, fossils, or holes may occur in marbles. As shown in the relevant article, the different grades of granite are easier to distinguish from each other than the marble samples in our study. In the studies of Doğan and Akay and Selver et al., limestone samples from the Manisa region were used. Although their chemical structure is very similar, there are many differences between limestone and marble in terms of their origin and physical properties. Marble has a variety of colors compared to limestone and it is more difficult for experts and computer automation systems to distinguish marbles of different qualities than limestone.

CHAPTER FOUR
CONCLUSION

In this study, By using the images of marble slabs, a new model and approach have been introduced in classifying the quality of marble. The originality of our study and its contribution to the relevant literature, the use of a specially designed convolutional neural network model in multiple classifications of marble quality, the use of two different image processing methods (blur filter and two-dimensional linear separator filter) in a way to eliminate the overfitting problem for data enhancement, and the difficult classification of data belonging to six different quality classes can be summarized as obtaining the process with very high accuracy rates. The accuracy rates obtained by the convolutional neural network model and data enhancement methods developed in our study were much higher than the rates obtained by using other artificial learning algorithms. Besides, Quality control experts and managers of "Haz Marble - Marble, Natural Stone, Fixing System Industry and Trade Inc." stated that the accuracy rates obtained with the convolutional neural network model are very close to the marble quality classification made manually by the experts. Since a specific national or international standard (ISO, TSE, etc.) is not yet in force for marble quality control classification, quality control and other related processes in the marble sector are made with expert comments on this issue. Therefore, the results we obtained in our study were evaluated by the marble quality control experts of "Haz Marble - Marble, Natural Stone, Fixing System Industry and Trade Inc.", whose marble images were obtained. The model presented in this study, can be thought of as an alternative that can be used in an automated manner with information systems to replace the manual quality control and quality classification processes performed by humans in marble factories and enterprises. Because Industry 4.0 solutions and applications are spreading very effectively and rapidly in many countries of the world, including our country, our model has a contribution and importance in this respect.

The following items made great improvements to our model after trial and error processes for us to deal with overfitting and underfitting.

- Adding more data (two different filters, one of them mixed filter),

- Decreasing model complexity (two convolutional layer ,no pooling layer),
- Using small kernel size (15x15),
- Small padding size on convolutional layers,
- Using 0.2 dropout rates before flatten and 0.5 before classifier,
- Using small batch size (underfitting),
- 10-fold cross-validation to learn faster with low loss.

One of the works we are planning shortly, To present the method and model in this study as an information product and application package for industrial applications in marble enterprises. Our pre-trained convolutional neural network model and image pre-processing process used for data enhancement, It can be embedded in a single software package or a single application that can be run on electronic camera devices or smartphones. These devices will be able to assign the marble to the correct quality class by taking a photograph of each new marble slab without human intervention by placing it at appropriate points on the marble processing benches. In this system, since a pre-trained convolutional neural network model is used for marble images, there will be no waiting time for the training phase. However, it should be noted that a certain runtime requirement will arise for image pre-processing processes in the live system. Therefore, For the automated marble quality classification process to be operated in real-time systems and industrial applications in an acceptable time, the image pre-processing phase must be performed using highly effective and fast algorithms and technologies.

REFERENCES

Aha, D., Kibler, D., & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning,* 37-66.

Andrew, N. (2015). What data scientists should know about deep learning. *Extract Data Conference*. Retrieved November 24, 2015, from https://www.slideshare.net/ExtractConf/andrew-ng-chief-scientist-at-baidu.

Avcı, K. (2018). Two dimensional digital filter design using Kaiser-Hamming window structure and Huang transform and image enhancement application. *Journal of the Faculty of Engineering and Architecture of Gazi University,* 1459-1474.

Bengio, Y. (2019). *Learning deep architectures for AI*. USA: now Publishers Inc.

Bianconi, F et al. (2012). Automatic classification of granite tiles through colour and texture features. *Expert Systems with Applications*, 11212-11218.

Bianconi, F et al. (2015). *On comparing colour spaces from a performance perspective: Application to automated classification of polished natural stones*. Switzerland: Springer.

Bonnin, R. (n.d.). *Machine learning for developers.* Retrieved February 14, 2021, from https://www.oreilly.com/library/view/machine-learning-for/9781786469878/252b7560-e262-49c4-9c8f-5b78d2eec420.xhtml.

Brownlee, J. (2016). *What is deep learning?* Retrieved February 14, 2021, from https://Machinelearningmastery.Com/What-is-deep-learning.

Brownlee, J. (2019). *How do convolutional layers work in deep learning neural networks?* Retrieved February 14, 2021, from https://machinelearningmastery.com/what-is-deep-learning/.

Brownlee, J. (2020). *How to choose an optimization algorithm.* Retrieved February 14, 2021, from https://machinelearningmastery.com/tour-of-optimization-algorithms/.

Buduma, N., & Locascio, N. (2017). *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. U.S.A.: O'Reilly.

Chakraverty, S., Goel, A., & Misra, S. (2018). *Towards extensible and adaptable methods in computing*. Singapore: Springer.

Convolutional neural network. (2019). Retrieved February 14, 2021, from https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-:0-2.

Desphande, A. (2019). *A beginner's guide to understanding convolutional neural networks.* Retrieved February 14, 2021, from https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/.

Doğan, H., & Akay, O. (2010). Using AdaBoost classifiers in a hierarchical framework for classifying surface images of marble slabs. *Expert Systems with Applications,* 8814-8821.

Ferreira, A., & Giraldi, G. (2017). Convolutional Neural Network approaches to granite tiles classification. *Expert Systems with Applications*, 1-11.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. Cambridge, MA, USA: The MIT Press.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hanbay, K. (2020). Hyperspectral image classification using convolutional neural network and two dimensional complex Gabor transform. *Journal of the Faculty of Engineering and Architecture of Gazi University,* 443-456.

Image Filtering. (2019). Retrieved February 14, 2021, from https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html.

Karaali, I. (n.d.). *Realka/datapreparation*. Retrieved February 14, 2021, from https://github.com/realka/DataPreparation.

Karaca, Z. (2003). Quality control of marble blocks. *MERSEM 2003 IV. Marble Symposium,* 497-503.

Kathuria, A. (2020*). Intro to optimization in deep learning: momentum, rmsprop and adam.* Retrieved February 14, 2021, from https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/.

Krizhevsky A, Nair V, & Hinton G. (2019). *The CIFAR-10 dataset.*. Retrieved September 21, 2019, from https://www.cs.toronto.edu/~kriz/cifar.html.

Küçük, H., İ, E., & Balcı, K. (2019). Classification of neuromuscular diseases with artificial intelligence methods. *Journal of the Faculty of Engineering and Architecture of Gazi University,* 1725-1742.

LeCun, Y et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation,* 541-551.

LeCun Y, Cortes C, & Burges CJC. (2019). *The MNIST Database of handwritten digits.* Retrieved November 8, 2019, from http://yann.lecun.com/exdb/mnist/.

Martinez-Alajarin, J., Luis-Delgado, J., & Tomas-Balibrea, L. (2003). Classification of marble surfaces using wavelets. *IEE Electronics Letters*, 714-715.

Martinez-Alajarin, J., Luis-Delgado, J., & Tomas-Balibrea, L. (2005). Automatic system for quality based classification of marble textures. *IEEE Transactions on Systems,Man, and Cybernetics,* 488-497.

Morphological transformations. (n.d.). Retrieved February 14, 2021, from https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html.

Pençe, İ., & Çeşmeli, M. (2019). Deep Learning in marble slabs classification. *Scientific Journal of Mehmet Akif Ersoy University,* 21-26.

Rectifier (neural networks). (2021). Retrieved February 14, 2021, from https://en.wikipedia.org/wiki/Rectifier_(neural_networks).

Selver, M.A et al. (2009). Cascaded and hierarchical neural networks for classifying surface images of marble slabs. *IEEE Transactions on Systems, Man, and Cybernetics,* 426-439.

Selver, M.A et al. (2011). An automated industrial conveyor belt system using image processing and hierarchical clustering for classifying marble slabs. *Robotics and Computer-Integrated Manufacturing,* 164-176.

Softmax function. (2021). Retrieved February 14, 2021, from https://en.wikipedia.org/wiki/Softmax_function.

Şişeci, M., & Cetişli, B. (2012). Determination of classes in travertine slabs using K-means and fuzzy C-means clustering methods. *Süleyman Demirel University Journal of the Institute of Science,* 238-247.

Unser, M. (1986). Sum and difference histograms for texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 118-125.

Utku, A., & Doğru, İ. (2017). Permission based detection system for android malware. *Journal of the Faculty of Engineering and Architecture of Gazi University,* 1015-1024.

Yavuz, A., & Koca, M. (2003). The use of micritic limestone as building stone: case study of Akhisar beige marble in western Turkey. *Proceedings of industrial minerals and building stones,* Istanbul, Turkey, 277-281.

Zhang, W et al. (1990). Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied Optics*, 4790-4797.

Zulkifli, H. (2018, January 21). *Understanding Learning Rates and How It Improves Performance in Deep Learning.* Retrieved February 14, 2021, from https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10.

APPENDICES

Appendix 1: The power to classify Q3A quality with our model

label	predictions	probabilities	img
Q3A	Q3A	0.998	./Q3A/Q3A-350.jpg
Q3A	Q3A	0.9463	./Q3A/Q3A-351.jpg
Q3A	Q3A	0.9965	./Q3A/Q3A-352.jpg
Q3A	Q3A	1.0	./Q3A/Q3A-353.jpg
Q3A	Q3A	0.9733	./Q3A/Q3A-354.jpg
Q3A	Q3A	0.9993	./Q3A/Q3A-355.jpg
Q3A	Q3A	0.9978	./Q3A/Q3A-356.jpg
Q3A	Q3A	0.8268	./Q3A/Q3A-357.jpg
Q3A	Q3A	0.973	./Q3A/Q3A-358.jpg
Q3A	Q3A	0.9997	./Q3A/Q3A-359.jpg
Q3A	Q3A	0.6968	./Q3A/Q3A-360.jpg
Q3A	Q3A	0.7837	./Q3A/Q3A-361.jpg
Q3A	Q3A	0.8728	./Q3A/Q3A-362.jpg
Q3A	Q3A	0.9442	./Q3A/Q3A-363.jpg
Q3A	Q3A	0.9655	./Q3A/Q3A-364.jpg
Q3A	Q3A	0.9706	./Q3A/Q3A-365.jpg
Q3A	Q3A	0.9077	./Q3A/Q3A-366.jpg
Q3A	Q3C	0.8507	./Q3A/Q3A-367.jpg
Q3A	Q3A	0.9877	./Q3A/Q3A-368.jpg
Q3A	Q3A	0.9889	./Q3A/Q3A-369.jpg
Q3A	Q3A	0.6368	./Q3A/Q3A-370.jpg
Q3A	Q3A	0.9995	./Q3A/Q3A-371.jpg
Q3A	Q3A	0.9894	./Q3A/Q3A-372.jpg
Q3A	Q3A	0.8144	./Q3A/Q3A-373.jpg
Q3A	Q3B	0.775	./Q3A/Q3A-374.jpg
Q3A	Q3B	0.9993	./Q3A/Q3A-375.jpg
Q3A	Q3A	0.9355	./Q3A/Q3A-376.jpg
Q3A	Q3A	0.8769	./Q3A/Q3A-377.jpg
Q3A	Q3A	0.9463	./Q3A/Q3A-378.jpg
Q3A	Q3A	0.8824	./Q3A/Q3A-379.jpg
Q3A	Q3A	1.0	./Q3A/Q3A-380.jpg
Q3A	Q3A	0.983	./Q3A/Q3A-381.jpg
Q3A	Q3A	0.9998	./Q3A/Q3A-382.jpg
Q3A	Q3A	0.9879	./Q3A/Q3A-383.jpg
Q3A	Q3A	0.9733	./Q3A/Q3A-384.jpg
Q3A	Q3A	0.9124	./Q3A/Q3A-385.jpg
Q3A	Q3A	0.9957	./Q3A/Q3A-386.jpg
Q3A	Q3A	0.9692	./Q3A/Q3A-387.jpg
Q3A	Q3A	1.0	./Q3A/Q3A-388.jpg
Q3A	Q3A	1.0	./Q3A/Q3A-389.jpg
Q3A	Q3A	0.9956	./Q3A/Q3A-390.jpg
Q3A	Q3A	0.7707	./Q3A/Q3A-391.jpg

Weak light.

Appendix 2: The power to classify Q3B quality with our model

label	predictions	probabilities	img
Q3B	Q3B	0.9862	./Q3B/Q3B-351.jpg
Q3B	Q3B	0.982	./Q3B/Q3B-352.jpg
Q3B	Q3B	0.8781	./Q3B/Q3B-353.jpg
Q3B	Q3B	0.9995	./Q3B/Q3B-354.jpg
Q3B	Q3B	0.4689	./Q3B/Q3B-355.jpg
Q3B	Q3B	0.7062	./Q3B/Q3B-356.jpg
Q3B	Q3B	0.989	./Q3B/Q3B-357.jpg
Q3B	Q3B	0.9523	./Q3B/Q3B-358.jpg
Q3B	Q3B	0.8287	./Q3B/Q3B-359.jpg
Q3B	Q3B	0.9952	./Q3B/Q3B-360.jpg
Q3B	Q3A	0.8945	./Q3B/Q3B-361.jpg
Q3B	Q3B	0.9214	./Q3B/Q3B-362.jpg
Q3B	Q4A	0.7038	./Q3B/Q3B-363.jpg
Q3B	Q3B	0.9661	./Q3B/Q3B-364.jpg
Q3B	Q3B	0.4974	./Q3B/Q3B-365.jpg
Q3B	Q3B	0.956	./Q3B/Q3B-366.jpg
Q3B	Q3A	0.5457	./Q3B/Q3B-367.jpg
Q3B	Q3A	0.7011	./Q3B/Q3B-368.jpg
Q3B	Q3B	0.9628	./Q3B/Q3B-369.jpg
Q3B	Q3B	0.9999	./Q3B/Q3B-370.jpg
Q3B	Q3B	1.0	./Q3B/Q3B-371.jpg
Q3B	Q3B	0.9998	./Q3B/Q3B-372.jpg
Q3B	Q3A	0.9987	./Q3B/Q3B-373.jpg
Q3B	Q3B	1.0	./Q3B/Q3B-374.jpg
Q3B	Q3B	0.9947	./Q3B/Q3B-375.jpg
Q3B	Q3B	0.9177	./Q3B/Q3B-376.jpg
Q3B	Q3B	0.9998	./Q3B/Q3B-377.jpg
Q3B	Q3B	0.9998	./Q3B/Q3B-378.jpg
Q3B	Q3B	0.9738	./Q3B/Q3B-379.jpg
Q3B	Q3B	0.9811	./Q3B/Q3B-380.jpg
Q3B	Q3A	0.8498	./Q3B/Q3B-381.jpg
Q3B	Q3A	0.6458	./Q3B/Q3B-382.jpg
Q3B	Q3B	0.9998	./Q3B/Q3B-383.jpg
Q3B	Q3A	0.8632	./Q3B/Q3B-384.jpg
Q3B	Q3A	0.8714	./Q3B/Q3B-385.jpg
Q3B	Q3A	0.8573	./Q3B/Q3B-386.jpg
Q3B	Q3A	0.9057	./Q3B/Q3B-387.jpg
Q3B	Q3B	0.9646	./Q3B/Q3B-388.jpg
Q3B	Q3B	0.998	./Q3B/Q3B-389.jpg
Q3B	Q3B	0.9997	./Q3B/Q3B-390.jpg
Q3B	Q3C	0.8447	./Q3B/Q3B-391.jpg
Q3B	Q3B	0.9991	./Q3B/Q3B-392.jpg

Weak light.

Appendix 3: The power to classify Q3C quality with our model

label	predictions	probabilities	img
Q3C	Q4B	0.402	./Q3C/Q3C-360.jpg
Q3C	Q3A	0.522	./Q3C/Q3C-361.jpg
Q3C	Q3A	0.9994	./Q3C/Q3C-362.jpg
Q3C	Q3C	0.9652	./Q3C/Q3C-363.jpg
Q3C	Q3C	0.9881	./Q3C/Q3C-364.jpg
Q3C	Q3C	1.0	./Q3C/Q3C-365.jpg
Q3C	Q3B	0.5225	./Q3C/Q3C-366.jpg
Q3C	Q3C	0.9487	./Q3C/Q3C-367.jpg
Q3C	Q3C	1.0	./Q3C/Q3C-368.jpg
Q3C	Q3C	0.9928	./Q3C/Q3C-369.jpg
Q3C	Q3B	0.5947	./Q3C/Q3C-370.jpg
Q3C	Q3C	0.9997	./Q3C/Q3C-371.jpg
Q3C	Q3C	0.6859	./Q3C/Q3C-372.jpg
Q3C	Q4B	0.9107	./Q3C/Q3C-373.jpg
Q3C	Q3C	0.9145	./Q3C/Q3C-374.jpg
Q3C	Q3C	0.8066	./Q3C/Q3C-375.jpg
Q3C	Q3C	0.7508	./Q3C/Q3C-376.jpg
Q3C	Q3C	0.9999	./Q3C/Q3C-377.jpg
Q3C	Q3C	0.962	./Q3C/Q3C-378.jpg
Q3C	Q3A	0.6617	./Q3C/Q3C-379.jpg
Q3C	Q3C	0.9949	./Q3C/Q3C-380.jpg
Q3C	Q3B	0.8407	./Q3C/Q3C-381.jpg
Q3C	Q3C	0.9933	./Q3C/Q3C-382.jpg
Q3C	Q3B	0.932	./Q3C/Q3C-383.jpg
Q3C	Q3C	1.0	./Q3C/Q3C-384.jpg
Q3C	Q3A	0.8095	./Q3C/Q3C-385.jpg
Q3C	Q3A	0.9198	./Q3C/Q3C-386.jpg
Q3C	Q4B	0.7685	./Q3C/Q3C-387.jpg
Q3C	Q4B	0.7685	./Q3C/Q3C-388.jpg
Q3C	Q3C	0.9998	./Q3C/Q3C-389.jpg
Q3C	Q3C	1.0	./Q3C/Q3C-390.jpg
Q3C	Q4B	0.9994	./Q3C/Q3C-391.jpg
Q3C	Q3C	0.9998	./Q3C/Q3C-392.jpg
Q3C	Q3C	1.0	./Q3C/Q3C-393.jpg
Q3C	Q3C	0.9353	./Q3C/Q3C-394.jpg
Q3C	Q4B	0.7633	./Q3C/Q3C-395.jpg
Q3C	Q3C	0.3963	./Q3C/Q3C-396.jpg
Q3C	Q3C	0.7834	./Q3C/Q3C-397.jpg
Q3C	Q3C	0.9955	./Q3C/Q3C-398.jpg
Q3C	Q3A	0.5218	./Q3C/Q3C-399.jpg
Q3C	Q3C	1.0	./Q3C/Q3C-400.jpg
Q3C	Q3C	0.9985	./Q3C/Q3C-701.jpg

Vein and not homogenous.

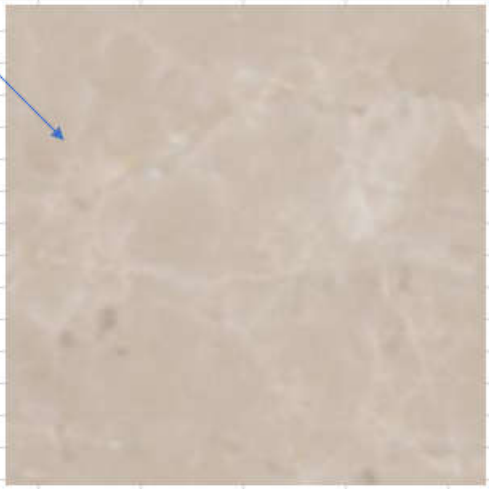
Appendix 4: The power to classify Q4A quality with our model

label	predictions	probabilities	img
Q3C	Q3A	0.5697	./Q3C/Q3C-351.jpg
Q3C	Q3C	0.9529	./Q3C/Q3C-352.jpg
Q3C	Q3B	0.9074	./Q3C/Q3C-353.jpg
Q3C	Q4B	0.9969	./Q3C/Q3C-354.jpg
Q3C	Q3C	0.9994	./Q3C/Q3C-355.jpg
Q3C	Q4C	0.9531	./Q3C/Q3C-356.jpg
Q3C	Q4B	1.0	./Q3C/Q3C-357.jpg
Q3C	Q3A	0.9994	./Q3C/Q3C-358.jpg
Q3C	Q3B	0.9893	./Q3C/Q3C-359.jpg
Q4A	Q4A	0.9941	./Q4A/Q4A-351.jpg
Q4A	Q4A	0.9763	./Q4A/Q4A-352.jpg
Q4A	Q4A	0.8492	./Q4A/Q4A-353.jpg
Q4A	Q4A	0.7816	./Q4A/Q4A-354.jpg
Q4A	Q4A	0.9115	./Q4A/Q4A-355.jpg
Q4A	Q4A	0.6787	./Q4A/Q4A-356.jpg
Q4A	Q4A	0.9404	./Q4A/Q4A-357.jpg
Q4A	Q4A	0.8723	./Q4A/Q4A-358.jpg
Q4A	Q4A	0.848	./Q4A/Q4A-359.jpg
Q4A	Q3C	0.9527	./Q4A/Q4A-360.jpg
Q4A	Q4A	0.9996	./Q4A/Q4A-361.jpg
Q4A	Q4A	0.9994	./Q4A/Q4A-362.jpg
Q4A	Q4A	0.9993	./Q4A/Q4A-363.jpg
Q4A	Q4B	1.0	./Q4A/Q4A-364.jpg
Q4A	Q4A	0.9988	./Q4A/Q4A-365.jpg
Q4A	Q3B	0.9841	./Q4A/Q4A-366.jpg
Q4A	Q4C	0.6924	./Q4A/Q4A-367.jpg
Q4A	Q4A	0.9424	./Q4A/Q4A-368.jpg
Q4A	Q4A	0.9931	./Q4A/Q4A-369.jpg
Q4A	Q4A	0.9798	./Q4A/Q4A-370.jpg
Q4A	Q4A	0.5912	./Q4A/Q4A-371.jpg
Q4A	Q4A	0.9976	./Q4A/Q4A-372.jpg
Q4A	Q4B	0.9978	./Q4A/Q4A-373.jpg
Q4A	Q4A	0.9986	./Q4A/Q4A-374.jpg
Q4A	Q4A	1.0	./Q4A/Q4A-375.jpg
Q4A	Q4A	0.7788	./Q4A/Q4A-376.jpg
Q4A	Q4A	1.0	./Q4A/Q4A-377.jpg
Q4A	Q4A	0.793	./Q4A/Q4A-378.jpg
Q4A	Q4A	0.4624	./Q4A/Q4A-379.jpg
Q4A	Q4A	0.5012	./Q4A/Q4A-380.jpg
Q4A	Q4C	0.8484	./Q4A/Q4A-381.jpg
Q4A	Q4A	0.8762	./Q4A/Q4A-382.jpg
Q4A	Q4A	0.8602	./Q4A/Q4A-383.jpg

Poor light in dark environment

Appendix 5: The power to classify Q4B quality with our model

label	predictions	probabilities	img
Q4B	Q4B	0.9965	./Q4B/Q4B-351.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-352.jpg
Q4B	Q4B	0.9999	./Q4B/Q4B-353.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-354.jpg
Q4B	Q4B	0.9821	./Q4B/Q4B-355.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-356.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-357.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-358.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-359.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-360.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-361.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-362.jpg
Q4B	Q4B	0.9973	./Q4B/Q4B-363.jpg
Q4B	Q4B	0.9999	./Q4B/Q4B-364.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-365.jpg
Q4B	Q4B	0.9829	./Q4B/Q4B-366.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-367.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-368.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-369.jpg
Q4B	Q4B	0.9918	./Q4B/Q4B-370.jpg
Q4B	Q4B	0.979	./Q4B/Q4B-371.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-372.jpg
Q4B	Q4B	0.5179	./Q4B/Q4B-373.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-374.jpg
Q4B	Q4C	0.9998	./Q4B/Q4B-375.jpg
Q4B	Q4B	0.9847	./Q4B/Q4B-376.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-377.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-378.jpg
Q4B	Q4C	0.9074	./Q4B/Q4B-379.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-380.jpg
Q4B	Q4C	0.7889	./Q4B/Q4B-381.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-382.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-383.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-384.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-385.jpg
Q4B	Q4C	0.8839	./Q4B/Q4B-386.jpg
Q4B	Q3C	0.9985	./Q4B/Q4B-387.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-388.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-389.jpg
Q4B	Q4B	0.9995	./Q4B/Q4B-390.jpg
Q4B	Q3C	0.7359	./Q4B/Q4B-391.jpg
Q4B	Q4B	1.0	./Q4B/Q4B-392.jpg



Appendix 6: The power to classify Q4C quality with our model

label	predictions	probabilities	img
Q4C	Q3B	0.9892	./Q4C/Q4C-381.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-382.jpg
Q4C	Q4C	0.9745	./Q4C/Q4C-383.jpg
Q4C	Q4B	0.7437	./Q4C/Q4C-384.jpg
Q4C	Q4C	0.8289	./Q4C/Q4C-385.jpg
Q4C	Q4A	0.8478	./Q4C/Q4C-386.jpg
Q4C	Q3A	0.9153	./Q4C/Q4C-387.jpg
Q4C	Q4A	0.8936	./Q4C/Q4C-388.jpg
Q4C	Q4A	0.5528	./Q4C/Q4C-389.jpg
Q4C	Q4C	0.9928	./Q4C/Q4C-390.jpg
Q4C	Q4B	0.9986	./Q4C/Q4C-391.jpg
Q4C	Q4C	0.9889	./Q4C/Q4C-392.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-393.jpg
Q4C	Q4C	0.9995	./Q4C/Q4C-394.jpg
Q4C	Q4B	1.0	./Q4C/Q4C-395.jpg
Q4C	Q4C	0.9773	./Q4C/Q4C-396.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-397.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-398.jpg
Q4C	Q4A	0.4272	./Q4C/Q4C-399.jpg
Q4C	Q4B	0.9895	./Q4C/Q4C-400.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-701.jpg
Q4C	Q4B	0.8059	./Q4C/Q4C-702.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-703.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-704.jpg
Q4C	Q4C	0.9419	./Q4C/Q4C-705.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-706.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-707.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-708.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-709.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-710.jpg
Q4C	Q4C	0.9693	./Q4C/Q4C-711.jpg
Q4C	Q4C	1.0	./Q4C/Q4C-712.jpg
Q4C	Q4C	0.9983	./Q4C/Q4C-713.jpg
Q4C	Q4C	0.9908	./Q4C/Q4C-714.jpg
Q4C	Q4C	0.7595	./Q4C/Q4C-715.jpg
Q4C	Q4C	0.9993	./Q4C/Q4C-716.jpg
Q4C	Q4C	0.9999	./Q4C/Q4C-717.jpg
Q4C	Q4C	0.5836	./Q4C/Q4C-718.jpg
Q4C	Q4C	0.9653	./Q4C/Q4C-719.jpg
Q4C	Q4B	1.0	./Q4C/Q4C-720.jpg
Q4C	Q4C	0.9999	./Q4C/Q4C-721.jpg
Q4C	Q4C	0.9899	./Q4C/Q4C-722.jpg

Poor light in dark environment