University of Tehran

Electrical and Computer Engineering Department

HW4 Machine Vision

Amin Fadaeinejad 810195442

Fall 2020

Abstract

In this computer assignment, we will learn more about the color space of an image, and what does every feature of a color space mean. After that we will use neural networks for classification,and we will see what will happen if we change the hyper parameters. In the end we will design networks with convolutional layers to classify the data-set.

1. (a) To visualize XYZ, think of a three-dimensional cartesian coordinate system (high school algebra) with axes labelled X, Y, and Z. In the XYZ color space, Y corresponds to relative luminance; Y also carries color information related to the eye's "M" (yellow-green) cone response. X and Z carry additional information about how the cones in the human eye respond to light waves of varying frequencies.

   There are some colour spaces that do cover all "visible" colours, but we wouldn't normally use them for images/videos. For example, that chart in your question shows colours in the CIE 1931 XYZ space, which is a colour space that covers all colours visible to humans (according to its psychological model).

   However, CIE XYZ is not a colour space that would normally be used to actually represent colour data, say in an image or video. The conversion back to an RGB space is relatively complex, it would waste a lot of bits of precision on space outside the range of colours most monitors can produce or sensors can see, even colours outside the space that humans can see. Mathematical operations that are simple to calculate in an RGB space would be highly complex in something like CIE XYZ and in all practicalities would require intermediate conversion anyway.

   An RGB colour space makes certain operations a lot easier. Monitors and screens use RGB colour spaces natively. If you're using an RGB colour space because your output medium is inherently RGB based, it initially makes sense to use a colour space that equals or closely matches the red, green and blue primaries that your output medium can do. In the past, colour monitors used phosphors that produced similar red, green and blue primaries, so that RGB space just because the "standard" colour space. Monitors are not all equal, increasingly so, and so inventing a device-independent colour space is a good idea: sRGB is the most common device independent space and it closely matches typical red, green and blue primaries from the CRT monitor era. sRGB has become a de facto standard for monitors, televisions (rec 601 and rec 709, used in digital video, pretty much reproduce it), and now the web and operating systems in general.

   So part of sRGB's popularity is its entrenchment in all those areas. As far as colour spaces go, and even as far as just RGB spaces go, it's very limited, and so you get Adobe RGB, ProPhoto, and the other RGB spaces with expanded gamuts. Encoding in them becomes just a little less efficient, necessitating the use of greater than 8 bits per channel in some cases, but they cover a wider gamut that new monitors and display technologies can do, and address a need for a "working colour space", where your input and output colour space can vary according to the device so you may as well use an intermediate space with a really wide gamut so it can convert between them with minimal loss. ProPhoto RGB,

often used as a ”working” colour space because it’s ”wide enough” to exceed just about any device colour space you can practically imagine, can cover almost all of the visible colours (according to CIE 1931) with the exception of some super deep greens and violets (again, these are far outside what monitors or other devices can display), but as a result it’s fairly inefficient to encode, with many coordinates simply not utilised because they fall outside the range of visible colours. Interestingly its primaries (ie its red, green and blue) are ”imaginary” - it is impossible to produce an emitter or sensor with the primaries of ProPhoto RGB because its primaries are impossible colours - they exist mathematically only, as a way to transfer colours to or from other spaces.

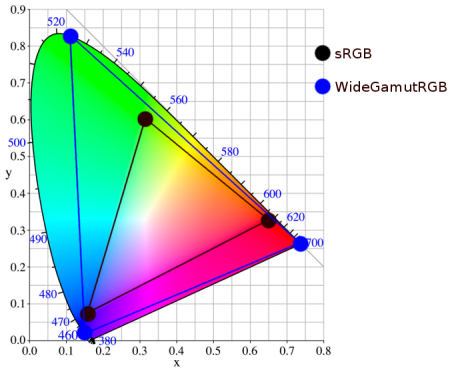

Figure 1: Color space

Figure 1 shows the color space, as we can see there are places where we can not represent in our color space.

(b) In this part we are going to compare the RGB and the sRGB color space with each other and plot every channel of the RGB color space.

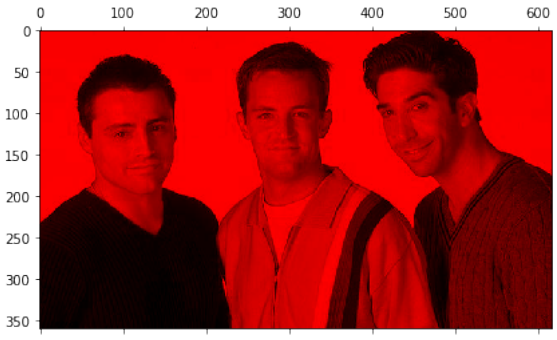

Figure 2: Red Channel

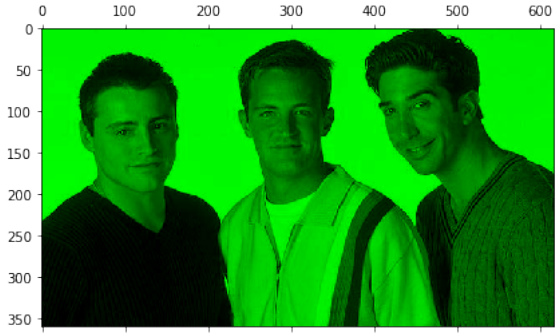

Figure 3: Red Channel

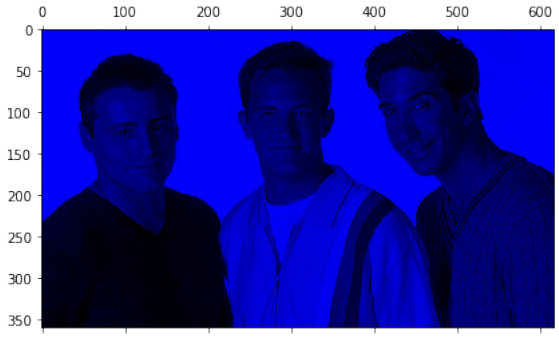

Figure 4: Red Channel

Figure 2,3,4 are the three color channels of the given image.

sRGB and Adobe RGB are two different color space profiles (also known as color models, or color systems). A color space is a range of possible colors. Depending on which mode you shoot in, your camera will capture a certain percentage of all visible colors. Some color models are better for print-based work. Others are more equipped for web-based media, guaranteeing consistency in appearance regardless of device or platform.

sRGB

Developed in 1996 through a joint effort from Microsoft and HP, sRGB emerged at an opportune time. The Internet was taking off; finally, everyday consumers were able to afford not just a personal computer of their own, but access to the web, too. At that point, sRGB was far and away the best model available, so

software companies relied on the color space when building and marketing new technology products (including the displays through which users would interact with the device).
It didn't take long before sRGB became the standardized color space option. (The S stands for standard, after all.)
That doesn't mean sRGB was the only color space profile available. At the time of its creation, there were dozens upon dozens of color space modes. But the industry needed to standardize, and sRGB won that round fair and square. Think of sRGB as a happy medium; to ensure colors are represented well in a consistent manner, sRGB will make subtle compromises. In layman's terms, through sRGB, colors will be dummied down, just a bit. Just enough so that, regardless of which monitor, device, or display type is being used, the images still look good—and the same. It should be noted that this simplification process isn't quite as dramatic as it sounds. The degree to which sRGB dampens colors is barely visible, even to the trained eye.
Consistency is really sRGB's bread and butter. Given the numerous different platforms used by people around the world, consistency in appearance is tremendously important. When shooting in sRGB, photographers needn't worry about how their work is represented online. What you see is what you will get, plain and simple.

Adobe RGB

To understand Adobe RGB and why it was developed, you must first understand another color model: CMYK. RGB is an additive color model, based on red, green, and blue, whereby different combinations will yield every color in existence. Or, if you mix everything together, you'll end up with white. It's the standard color model for the web, given that laptops, tablets, and smartphones have rich displays capable of rendering each color. CMYK, however, is a subtractive color model composed of cyan, magenta, yellow, and key (black). Colors are formed by deducting hues from another. The entire spectrum will ultimately yield black, rather than white.
RGB works best in digital environments. But, in the physical world where digital creations are brought to life via printing, it won't always boast the luster it has on your screen—hence why most print shops operate exclusively in CMYK. CMYK offers greater fidelity to the color and the integrity of the original piece of work. That way, print companies are not liable if a customer's photo comes out duller than expected. While sRGB assures consistency for photographers regardless of the platform, CMYK does the same for prints.
That idea—pandering to CMYK printers and which colors they're able to achieve—

formed the basis of Adobe RGB, which was created in 1998. Adobe Systems sought to provide photographers with an alternative option for shooting and presenting their work, while also expanding upon the list of visible colors it can capture.
With a wider gamut of colors, Adobe RGB, in theory, offers greater potential than sRGB. But with great potential comes greater challenges; working exclusively in Adobe RGB vs. sRGB requires additional legwork, particularly during post-processing. If you shoot in Adobe RGB and like to retouch your photos, editing will entail a few extra steps—something that's especially true if you hope to share your work online in your online photography portfolio. You can indeed convert to sRGB from Adobe RGB, and with decent results, too. But it won't look nearly as good if you had originally shot them in sRGB. If you import Adobe RGB images into an sRGB work environment, the program will convert the colors accordingly, which is often overlooked by the user. The same goes for exporting; an Adobe RGB workflow means you have to dot your i's and cross your t's. Miss a step and your work will suffer.

Result

Until support is strengthened, sRGB is far and away the best color space option available to photographers. The color gamut may not be as rich as Adobe RGB, but, to the naked eye, such differences are marginal. There's a reason sRGB has been the default color space since the dawn of the modern web: it's universally supported. Adobe RGB, while powerful, will always add a few extra steps to your workflow. It will always require more post-processing work than sRGB. It will always present additional challenges to overcome, which can detract from the root of your photographic work.
Adobe RGB is also one of the leading causes of substandard prints. Yes, when done right, Adobe RGB can produce better results for print. All too often, however, great photographers who are not so technically gifted will find their prints don't do justice to the original work. With Adobe RGB, the colors will not always match what you see on your monitor, effectively reducing the quality of your work. sRGB is the default color space; context aside, you can be assured the colors will be represented accurately. Adobe RGB poses too many potential headaches to earn the title of color space king. For the time being, sRGB the best color space available. Photographers want their work to be viewed and appreciated as they intended. Whether you're shooting in sRGB or Adobe RGB, only the former can safeguard your vision—only sRGB can enable you to take the best photographs possible.

(c) In this part we are going to convert the image in the RGB domain to HSV domain using the function that we have implemented.

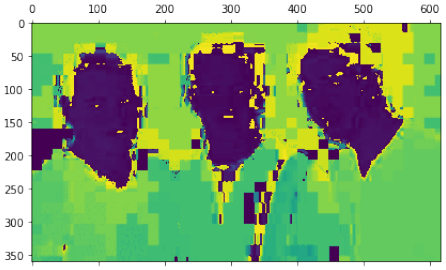

Figure 5: H channel

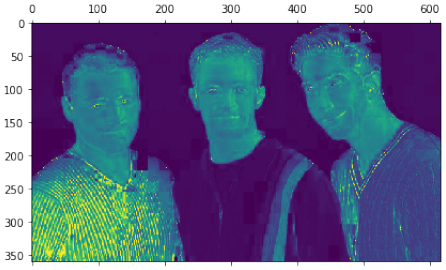

Figure 6: S channel

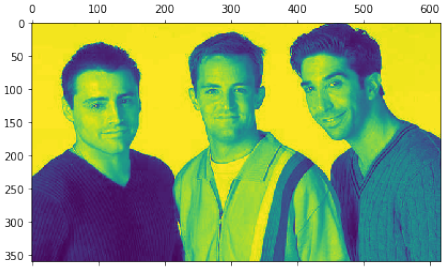

Figure 7: V channel

The HSV color wheel sometimes appears as a cone or cylinder, but always with these three components:

Hue

Hue is the color portion of the model, expressed as a number from 0 to 360 degrees:
Red falls between 0 and 60 degrees.
Yellow falls between 61 and 120 degrees.
Green falls between 121 and 180 degrees.
Cyan falls between 181 and 240 degrees.
Blue falls between 241 and 300 degrees.
Magenta falls between 301 and 360 degrees.

Saturation

Saturation describes the amount of gray in a particular color, from 0 to 100 percent. Reducing this component toward zero introduces more gray and produces a faded effect. Sometimes, saturation appears as a range from 0 to 1, where 0 is gray, and 1 is a primary color.

VALUE (OR BRIGHTNESS)

Value works in conjunction with saturation and describes the brightness or intensity of the color, from 0 to 100 percent, where 0 is completely black, and 100 is the brightest and reveals the most color.

(d) In this part we are going to convert the RGB image to the YCbCr color space.

Figure 8: Y channel

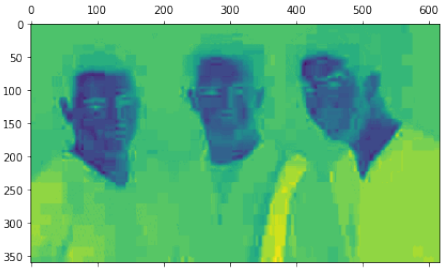

Figure 9: Cb channel

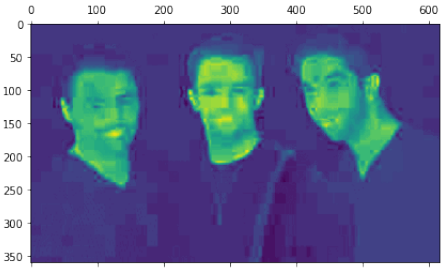

Figure 10: Cr channel

YCbCr, Y CbCr, or Y Pb/Cb Pr/Cr, also written as YCBCR or Y CBCR, is a family of color spaces used as a part of the color image pipeline in video and digital photography systems. Y is the luma component and CB and CR are the blue-difference and red-difference chroma components. Y (with prime) is distinguished from Y, which is luminance, meaning that light intensity is non-linearly encoded based on gamma corrected RGB primaries.

$$
\begin{aligned}
Y' &= K_R \cdot R' + K_G \cdot G' + K_B \cdot B' \\
P_B &= \frac{1}{2} \cdot \frac{B' - Y'}{1 - K_B} \\
P_R &= \frac{1}{2} \cdot \frac{R' - Y'}{1 - K_R}
\end{aligned}
$$

Figure 11: Formula

Y CbCr color spaces are defined by a mathematical coordinate transformation from an associated RGB color space. If the underlying RGB color space is ab-

solute, the Y CbCr color space is an absolute color space as well; conversely, if the RGB space is ill-defined, so is Y CbCr.
YCbCr is sometimes abbreviated to YCC. Y CbCr is often called YPbPr when used for analog component video, although the term Y CbCr is commonly used for both systems, with or without the prime.
Y CbCr is often confused with the YUV color space, and typically the terms YCbCr and YUV are used interchangeably, leading to some confusion. The main difference is that YUV is analog and YCbCr is digital.
Y CbCr signals (prior to scaling and offsets to place the signals into digital form) are called YPbPr, and are created from the corresponding gamma-adjusted RGB (red, green and blue) source using three defined constants KR, KG, and KB as follows:

(e) Average method is the most simple one. You just have to take the average of three colors. Since its an RGB image, so it means that you have add r with g with b and then divide it by 3 to get your desired grayscale image.
There is one thing to be sure, that something happens to the original works. It means that our average method works. But the results were not as expected. We wanted to convert the image into a grayscale, but this turned out to be a rather black image.
This problem arise due to the fact, that we take average of the three colors. Since the three different colors have three different wavelength and have their own contribution in the formation of image, so we have to take average according to their contribution, not done it averagely using average method. Right now what we are doing is this,33% of Red, 33% of Green, 33% of Blue We are taking 33% of each, that means, each of the portion has same contribution in the image. But in reality thats not the case. The solution to this has been given by luminosity method.
You have seen the problem that occur in the average method. Weighted method has a solution to that problem. Since red color has more wavelength of all the three colors, and green is the color that has not only less wavelength then red color but also green is the color that gives more soothing effect to the eyes.
It means that we have to decrease the contribution of red color, and increase the contribution of the green color, and put blue color contribution in between these two.
New grayscale image = ((0.3 * R) + (0.59 * G) + (0.11 * B)).
According to this equation, Red has contribute 30%, Green has contributed 59% which is greater in all three colors and Blue has contributed 11%.

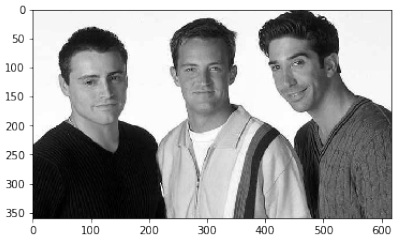

Figure 12: Grey Image

(f) In this part we are going to find a mask that can represent the face of the three boys. We first normalized the image points and after that we will apply a mask where the normalized points are in a particular range they will get masked and if they are not in range they will get rejected.

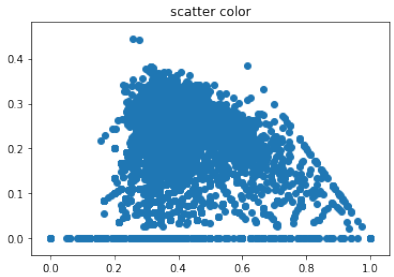

Figure 13: The scatter plot of the color region

Figure 13 shows the scatter plot of the color space.

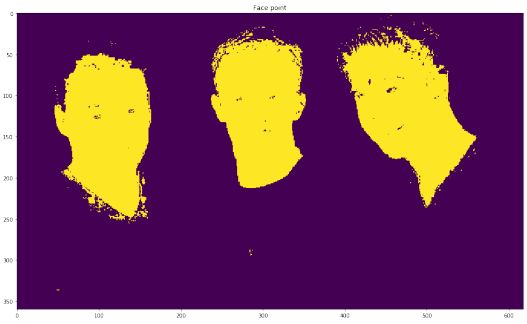

Figure 14: Face

Figure 14 shows that mask that we can apply to the image.

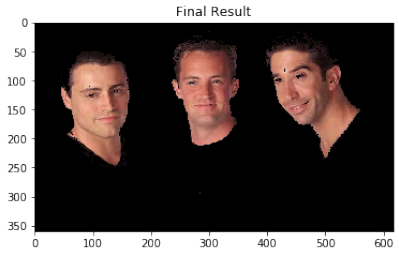

Figure 15: Grey Image

Figure 15 shows the result after applying the mask.

2. In this question we are going to design a deep network from scratch.

   (a) In the first part we assume the initial value of the weights are hero, since the activation functions are Relu so if the all the inputs are 0 then there will be no gradient to backpropagate. And we achieved 10 % for the accuracy.

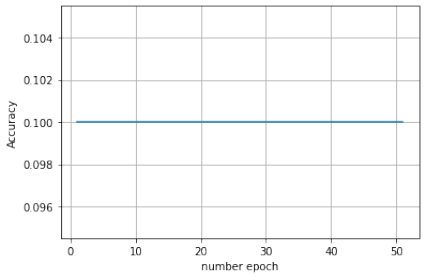

Figure 16: Accuracy

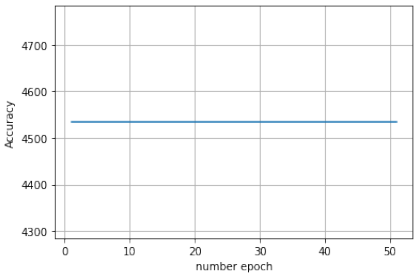

Figure 17: Loss

As we can see in figure 16 and 17 since all the weights are 0 there is going to be no change in the network.

We will change the method for initializing the weights, we used the mean of 0 and the STD of 0.1 and we got the best result. And we achieved 43 % for the accuracy.

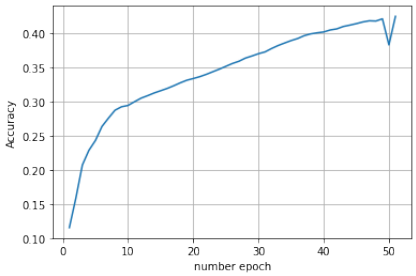

Figure 18: Accuracy Train

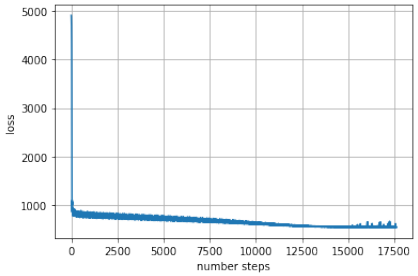

Figure 19: loss Train

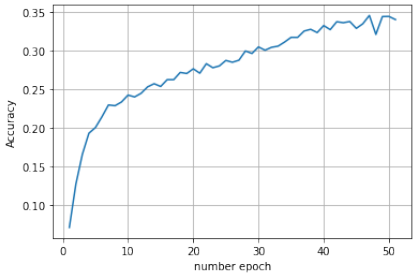

Figure 20: loss Train

(b) In this part we are going to try to implement this network with more hidden layers. We will once use 2 layers and the next time we will use 3 layers.

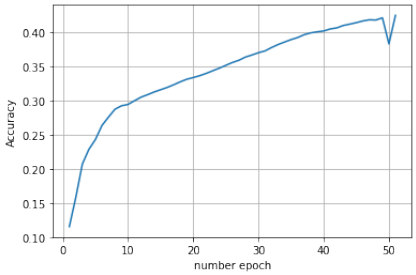

Figure 21: Accuracy Train

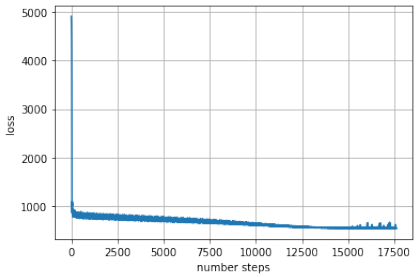

Figure 22: loss Train

Figures above shows the loss and accuracy of the 2 layer network. And we achieved 43 % for the accuracy.

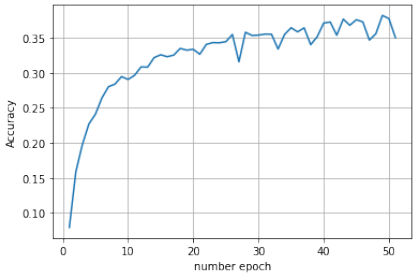

Figure 23: Accuracy Validation

The above figure shows accuracy of the validation data set.

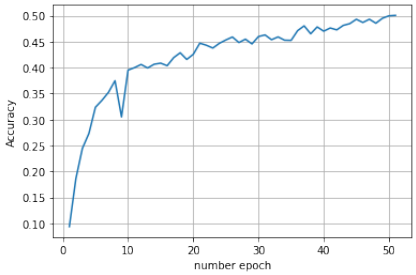

Figure 24: Accuracy Train

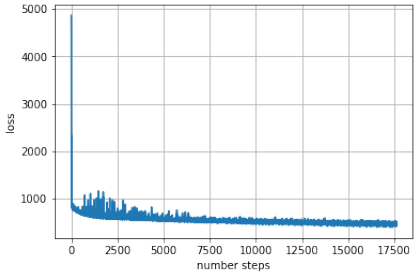

Figure 25: loss Train

Figures above shows the loss and accuracy of the 3 layer network.

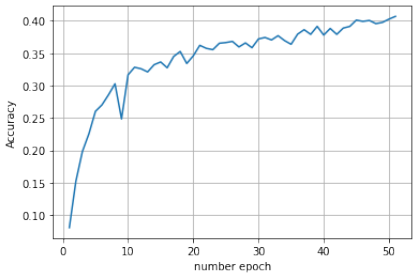

Figure 26: Accuracy Validation

The above figure shows accuracy of the validation data set. And we achieved 51 % for the accuracy for test.

(c) In this part we are going to change the number nodes in the hidden layer and we will see how does this impact the result that we want to achieve. And we achieved 30 % for the accuracy.

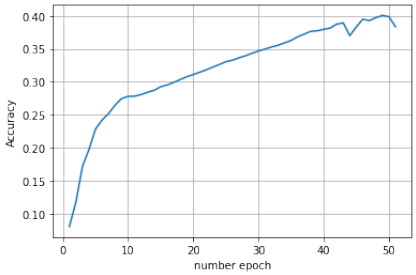

Figure 27: Accuracy Train

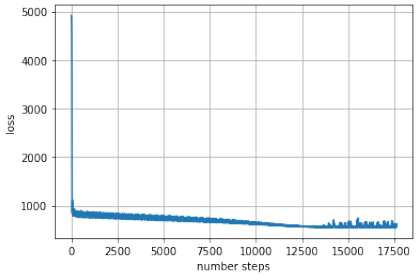

Figure 28: loss Train

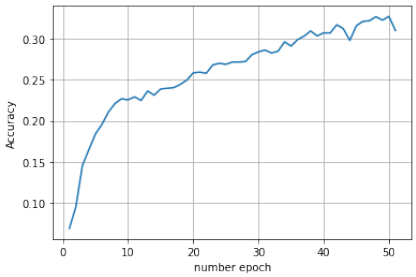

Figure 29: Accuracy validation

The figures above show the accuracy and loss of the train and validation for the hidden number node of 50. And we achieved 43 % for the accuracy.

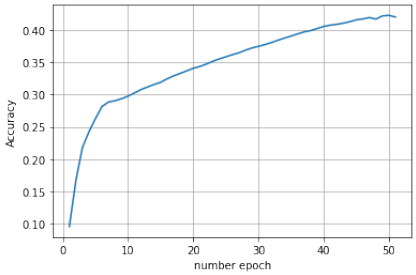

Figure 30: Accuracy Train

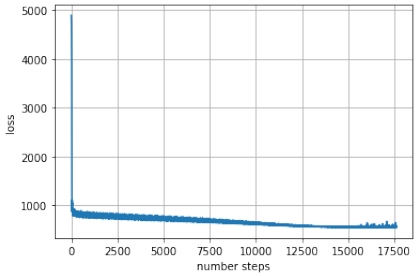

Figure 31: loss Train

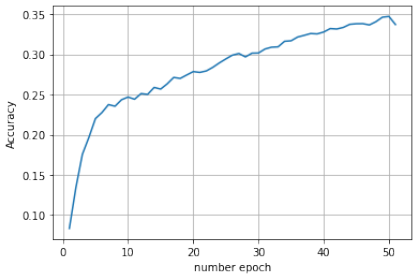

Figure 32: Accuracy validation

The figures above show the accuracy and loss of the train and validation for the hidden number node of 200.

(d) In this part we are going to change the learning rate to see how does it impact the result of the network. In this part we are going to use 0.01 as the learning rate. And we achieved 27.8 % for the accuracy.

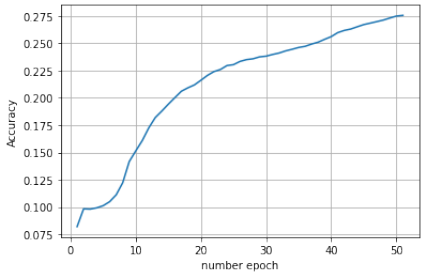

Figure 33: Accuracy Train

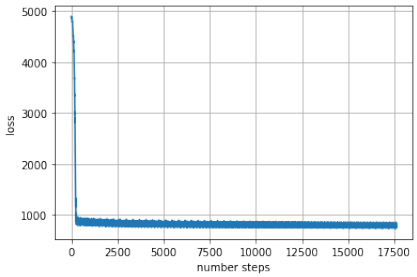

Figure 34: loss Train

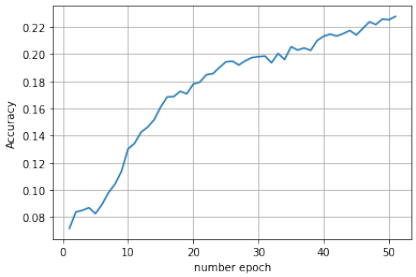

Figure 35: Accuracy validation

The following figures belong to the network with the learning rate of 0.0001. And we achieved 27.5 % for the accuracy.

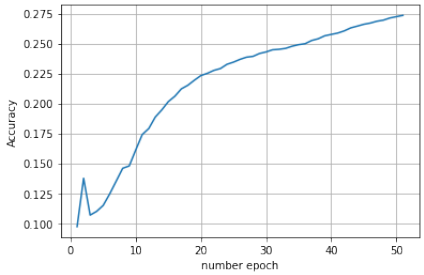

Figure 36: Accuracy Train

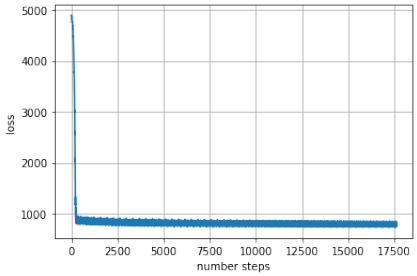

Figure 37: loss Train

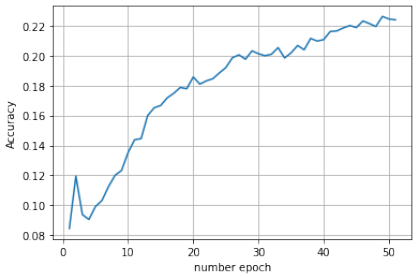

Figure 38: Accuracy validation

(e) In this part we are going to use other methods in initialize the network but since the best result came from the method we are mean is 0 and STD is 0.1 we assumed that at finished the previous part with this assumption. And we tried to use zero as the initial value but all the results were 0. We also used to uniform distribution for the initializing and got these result. And we achieved 20 % for the accuracy for the uniform distribution and 10 % for the zeros initialization method.

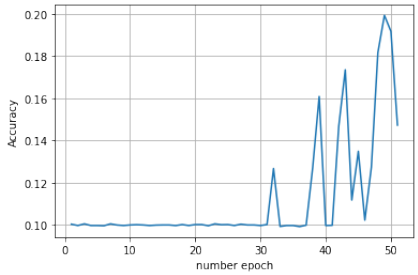

Figure 39: Accuracy Train

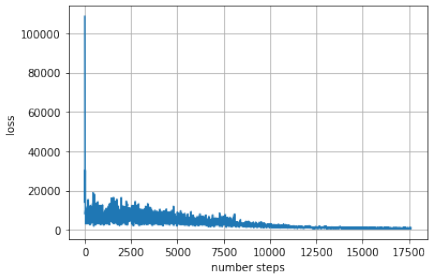

Figure 40: loss Train

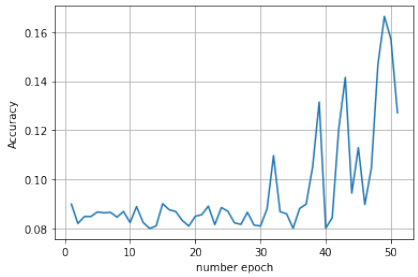

Figure 41: Accuracy validation

As you can see in the figures we can understand that

3. In this part we are going to complete the bonus parts.

   (a) In this part we are going to apply the PCA method to our data then we will use the previous network for the job, the result are like the following. And we achieved 42 % for the accuracy.

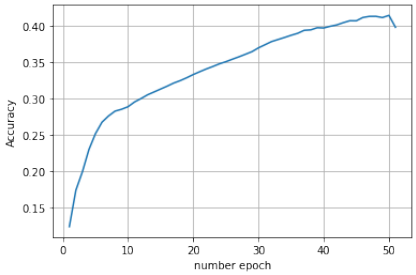

Figure 42: Accuracy Train

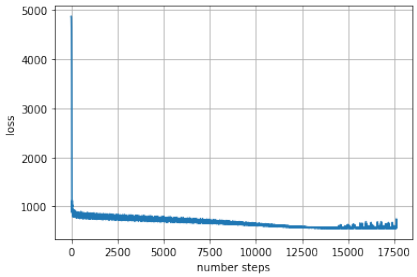

Figure 43: loss Train

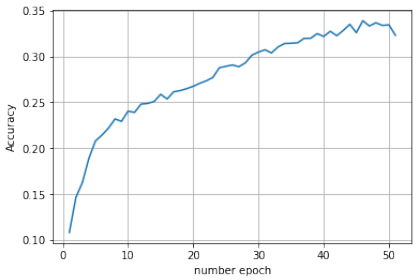

Figure 44: Accuracy validation

(b) In this part we are going to use the gray scale image to classify the image. And we achieved 33 % for the accuracy.

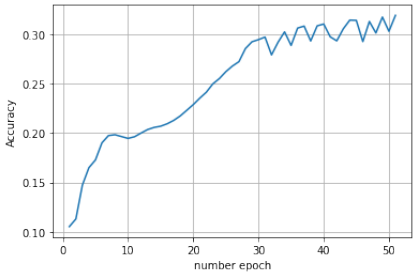

Figure 45: Accuracy Train

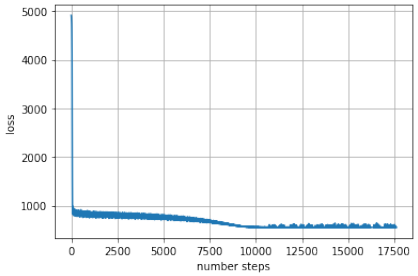

Figure 46: loss Train

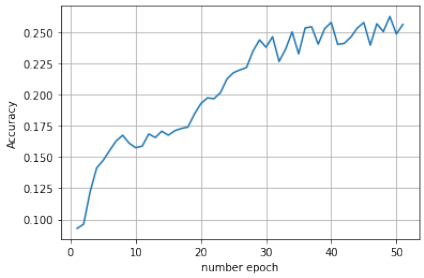

Figure 47: Accuracy validation

(c) In this part we are going to jump to a conclusion using PCA or gray scale. According to the accuracy we can see that the best case is when we use the PCA we will get the maximum accuracy, and the bigger the network the more accuracy we have (this applies for the number of nodes and). So we can find out the best result is for the higher learning rate(since we trained it for 50 epoch).

4. In this part we are going to implement a neural network with the tensor flow library.

(a) In the first part we are going implement a model with conv layers added to the last model. The accuracy is 97 %.

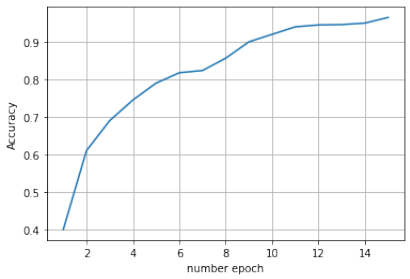

Figure 48: Accuracy Train

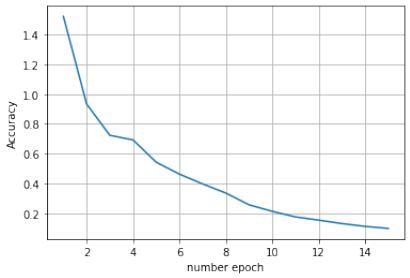

Figure 49: loss Train

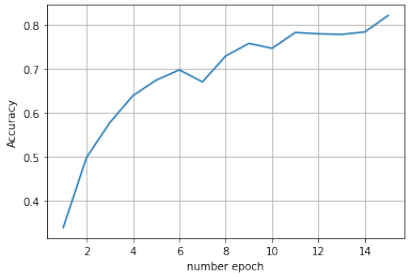

Figure 50: Accuracy validation

(b) In this part we are going to show some of the conv layers in the network to come up with a sense towards them. We show the image of a number of layers in 3 stages one in the first epoch one in the 7th epoch and one in th 14th epoch.

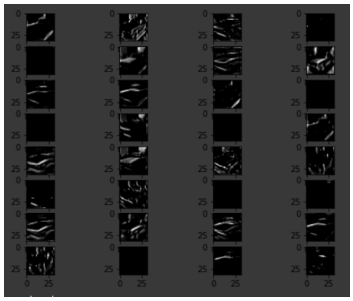

Figure 51: Epoch 1

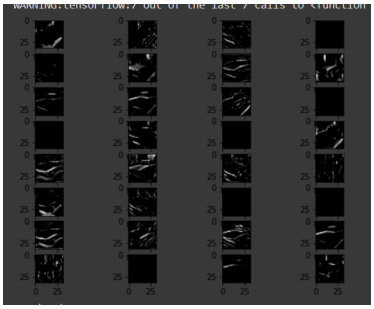

Figure 52: Epoch 7

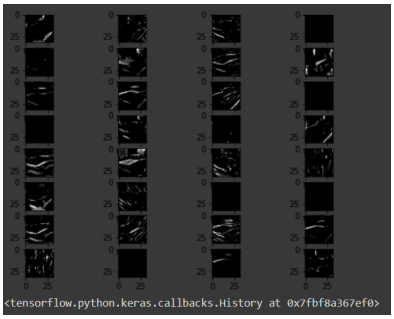

Figure 53: Epoch 14

(c) In this part we will use the pooling layers in the network and will plot the result. The accuracy is 95 %.

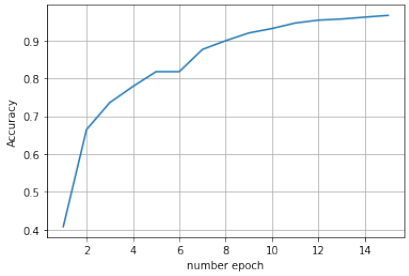

Figure 54: Accuracy Train

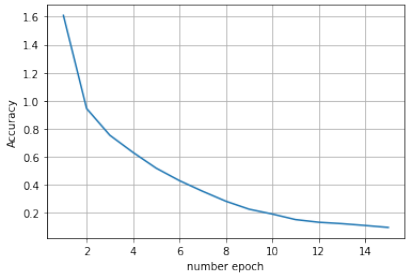

Figure 55: loss Train

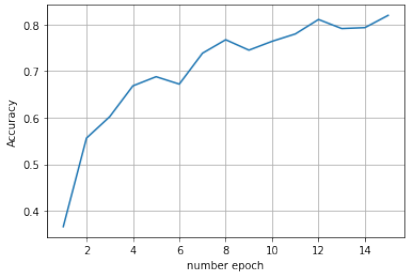

Figure 56: Accuracy validation

As we can see in the figures the accuracy has decreased in this time. So overall the pooling is a way we can reduce the number of parameters, and by reducing the parameters the network will train faster.

Reference

1. Why don't color spaces use up the entire color spectrum?
2. how-to-get-the-output-of-all-layers-of-a-keras-model

3. Softmax function
4. softmax-crossentropy
