Computer Vision 1

Shrey Gupta & Karthik Bhargav & Tiya Singh

February 26, 2021

What is Computer Vision?

Computer vision is a field in artificial intelligence where the machine learns to understand features in images. As the name itself, computer vision is used to teach a computer how to interpret vision and visual images.

Applications

Computer vision is used in many different applications including medical image analysis, face recognition, autonomous driving, augmented reality, unmanned vehicles, and more. It can be used as a tool to detect diseases through x-rays or MRI scans for healthcare professionals. Below you can see an image of a model detecting pneumonia through a CXR image.

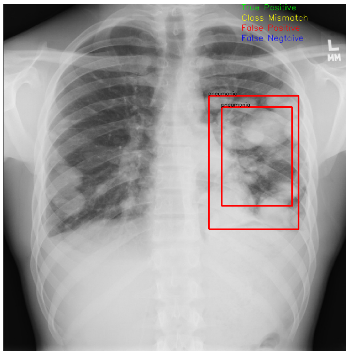


Figure 1: Model detecting pneumonia

Computer vision is also used in creating self-driving cars in order to detect traffic lights, other cars, pedestrians, etc. Below is another application of computer vision in facial recognition, finding faces in a picture.

Challenges

While computer vision may be used in many different applications, it does come with its challenges. One of them being the fact of ethical considerations.

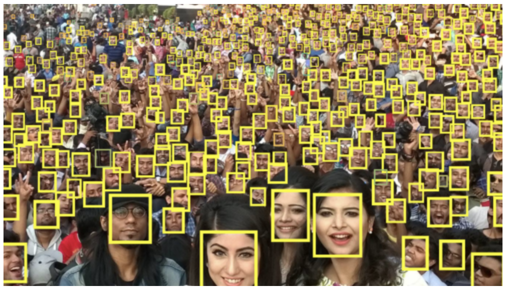


Figure 2: Model detecting pneumonia

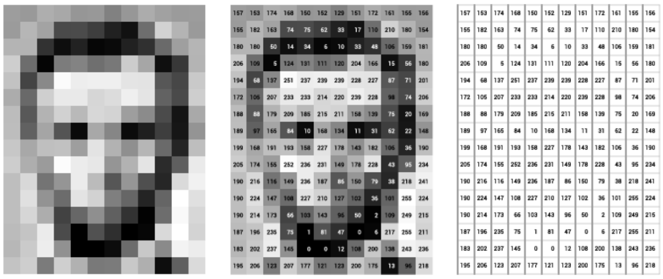
A model's accuracy and performance is dependent on how well it trains on the dataset. While a model may have trained very well on a dataset of detecting white hands, it might perform worse on detecting hands of different races. With such a diverse race and ethics of people, the model cannot be perfect in detecting everything causing it to be a challenge yet to overcome. With every product of great use, in the wrong hands it can be used for bad things as well. An example of this is deepfakes. Deepfakes are referred to when there is a form of manipulation done on images or videos to turn into different, newer representations of that image. Unfortunately this simple algorithm can be used in bad ways, such as pretending to be someone, discrediting a public figure, impersonating someone, and more. This is another dangerous challenge we have to overcome in the field of computer vision.

Future Growth

The computer vision market is expected to reach 25.32 billion US dollars by 2023. Image recognition and computer vision techniques offer a lot of significant opportunities for industry growth in this field. The automotive computer vision industry expects to grow fast and by a lot, thought to be the largest end-user of computer vision applications. This field is expected to reach around 31.65% CAGR in a forecasting period from 2013 to 2023. Computer vision career possibilities range from small business jobs to computer vision based companies. Image recognition engineers are a possible career with this knowledge and experience. Research is also another pathway career if one is strong in this computer vision field. They would be working on projects related to image recognition.

Interpreting

The brain is able to process visual information in front of us and detect different objects, track motion, and specifically distinguish between objects of the same type. A popular paper written by James J. DiCarlo, Davide Zoccolan, and Nicole C. Rust suggests that the brain uses patterns to decipher what our eyes process. Computer vision works the same way as it uses patterns to interpret what data we feed it. For example, if we provide our program with thousands of pictures of white, brown, and black bears, the computer will attempt to find patterns in those images that are unique and similar to white, brown, and black bears. If you were to give this program a picture of a spoon, we would hope that our model does not think that it resembles a bear, although unanticipated things like this can happen.



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Figure 3: Image matrix

The image above shows how computer vision works at its core. Each image is composed of pixels, covered in more detail below, which are assigned values. These values can be based on a variety of factors including color, intensity, etc. Each pixel is then stored inside of a 1-dimensional array. The computer looks at this array and is able to find patterns that it then saves and cross-references with other images to improve its pattern detection feature.

Pixels

In computer vision, the simplest stream of data is a single image. From this, images are often expressed in the form of a 2D array of pixels. Pixels are the smallest unit that can be displayed on a screen, and they are considered basic building blocks of an image. Pixels are also used to describe the a specific color or value at a single place in an image.

The resolution of an image is denoted as the ***number of pixels in the height*** x ***number of pixels in the width*** of an image. It is important to note that a pixel doesn't have a specific dimension rather it depends on the the pixels per inch(PPI) set for the image.

Pixels are represented by a range of numbers known as *color depth*, and these are then used to represent color in an image which will be discussed in the next section. Images contain a lot of data, and because of this there are various image compression algorithms used to efficiently store information about pixels, from images.

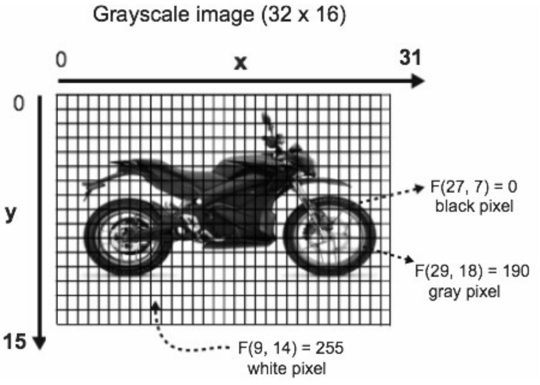


Figure 4: Image Size, and Pixel Color

Colors and Light

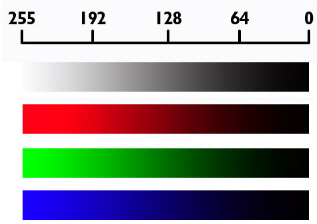


Figure 5: RGB and grayscale output from 0 to 255

Images are the input to computer vision models. They usually appear as colored or grayscale images. Every image is built through a grid of pixels, it is the building block of the images. Each pixel is a number ranging from 1 to 255 showing the intensity of that pixel. For example, a grayscale image with size 32x16 has 32*16 pixels. In colored images, the same rules are applied but for three different colored channels. Specifically there are three main colors that can change intensity value to create every possible color. These channel colors are red, green and blue. As an example, a colored image of size 32x16, has 32*16*3 pixels. In colored images, there is a term called "alpha channel" which refers to the channels and is used to define certain pixel intensity values. By adding an alpha channel of the color red we are essentially magnifying the red pixel values. It defines specific areas of the image. In summary, when machines read images, they look at an array of these pixel values which represent intensity values of the corresponding color.

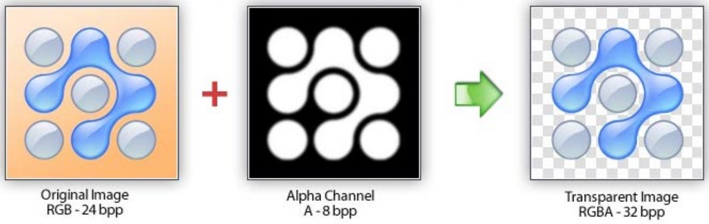


Figure 6: Image Size, and Pixel Color

Filters

Think of an image now as a matrix of pixel values that are represented by a function and filters are just different adjustments we make to this function to get more meaning out of the image. The reason computer scientists may want to use filters is to gather information about the image or enhance the various aspects of an image. Below is an example of super-resolution, a type of filter.

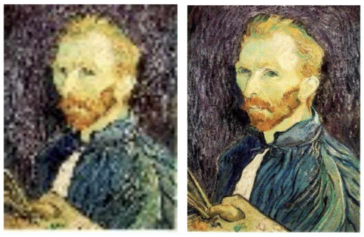
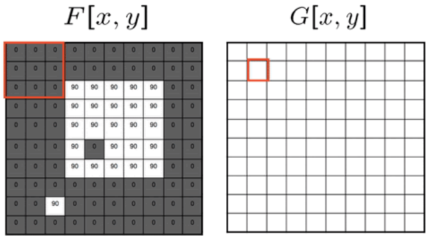


Figure 7: Before and after of the super resolution filter

Moving Average Filter and Image Segmentation

The moving average filter and the image segmentation filter are two commonly used filters. The moving average takes a pixel on the image and averages its value with its neighboring pixel values. It is best to think about a 3 by 3 box moving over an image and replacing the box's center pixel with the average value of the numbers inside the box. This box's formal name is called a kernel. A kernel is used in most filters and it is essentially another matrix that is multiplied by the matrix found in the image. The intensity of the filter can range as the average filter size can be different.

Below is an example of what the moving average filter can produce. This filter can potentially make edges on the image as the edge pixels would not be in the center of the box. To fix this, many methods can be employed. A common strategy is using zero padding which adds an n-width pixel border around the image with a value of 0.



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 8: Mathematical display of the moving average filter

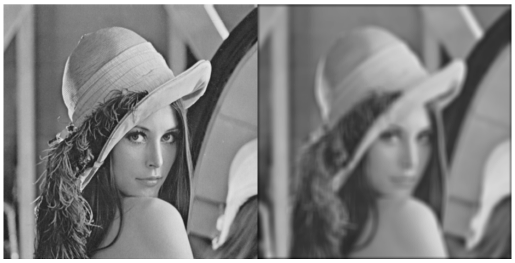


Figure 9: Before and after of the moving average filter

Image segmentation separates an image into different regions using a function in order to highlight boundaries or to easily identify objects. For example, if we use a function that combs every pixel and says if the pixel has a value over 127 then replace it with 255, max intensity, otherwise replace it with 0, min intensity. In the image below, it is evident how this filter highlighted the boundaries clearer.

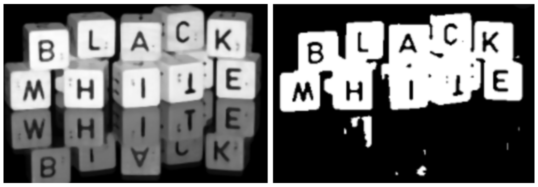


Figure 10: Before and after of a type of image segmentation

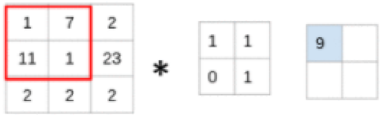
Other examples of filters are sharpening, blurring, de-noising and in-painting. These all fundamentally use the same concept of applying a kernel to an image, affecting the pixel values, creating a new, more meaningful image.

Convolutions

In short, a convolution is just an element-wise multiplication of two matrices followed by taking the sum of the elements of the resulting matrix. Although the definition itself is quite simple it's important to understand it's applications to CV in particular.

An image is a 2D array of pixels(as we discussed earlier). In terms of a traditional neural network, a convolution is similar, taking in the image and multiplying a set of weights on the input. This set of weights is a 2D array known as a filter or kernel.

The kernel is a small matrix that is used to slide across the image and perform the convolution operation described earlier. Depending on the size of an image the kernel size can range from (2x2) to (7x7). The larger the kernel size the less data we can extract from the image, however the smaller the kernel size the longer time it takes to perform the convolution. Let us take a look at an example of a convolution.



$$\begin{bmatrix} 1 & 7 & 2 \\ 11 & 1 & 23 \\ 2 & 2 & 2 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 9 & \\ & \end{bmatrix}$$

Figure 11: Example of a Convolution

As you can see from this example, each column in the image is multiplied by the kernel. From the resulting matrix we take the sum of the elements in the matrix. Afterwards, we put the resulting sum at the same coordinates as the center of the kernel. If the kernel sizes are even(as shown in the image) we take the coordinates of the top left corner of where the kernel currently is.

Different kernels can be used to extract different features from an image, and kernels can be used for a wide variety of things such as edge detection, and applying various types of blurs.

There are two techniques often used with convolutions, they are zero-padding and strides. Often times when using convolutions the center pixels have significantly more influence on the output compared to its surrounding edges. That is why by using zero-padding, an extra layer of pixels with a value of 0 are added to the original image. Stride size determines the step size of a filter across the image. By increasing the stride size the resulting output image is significantly reduced in size causing less information to be learned. By decreasing the stride size more information is learned from the image but takes a longer time(similar to kernel size)

To determine the dimensions of an image after a convolution the following formula is used.

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}:	number of input features
n_{out}:	number of output features
k:	convolution kernel size
p:	convolution padding size
s:	convolution stride size

Figure 12: Before and after of the moving average filter

The formula used is take the floor of (input dimensions + 2*(zero padding size) - (kernel size) / (the size of the stride)) + 1.

Using deep learning and convolutional neural networks(CNN's), models can learn the most optimal kernels for specific image recognition tasks such as classifying dogs and cats to more important problems such as classifying various types of diseases.

References

1. Image Filtering by Yeung, Serena
2. Pixels and Images by O'Reilly
3. Understanding Convolutions by Irhum Shafkat
4. Convolutions with OpenCV and Python by Adrian Rosebrock
5. What Is Computer Vision & How Does it Work? An Introduction