COMP 642: Machine Learning
Mar 2nd, 2023

Lecture 15: Featurization of Images and Convolution Neural Networks

---

Lecturer: Anshumali Shrivastava
Scribe By: Keith Pulmano, Matias Romero, Risto Trajanov, Sam Welsh

Spring Semester 2023

Contents

Disclaimer: *These lecture notes are intended to develop the thought process and intuition in machine learning. The materials are not thoroughly reviewed and can contain errors.*

1 Images in Machine Learning

So far, we have talked about how to deal with numbers and words in Machine learning models. In this segment of the course, we discuss images and how they are used in the subset of Machine Learning called Computer Vision. To begin, we must consider a key question: how can we feed an image into a regression or a classification model?

To answer this question, we discuss how a digital image is typically stored and a popular representation that allows images to be compatible with Machine Learning models.

1.1 Concept of an image

In a typical image file, three different color channels are stored separately (containing RGB values). One channel represents the red intensity, one that represents the green intensity, and one that represents the blue intensity.

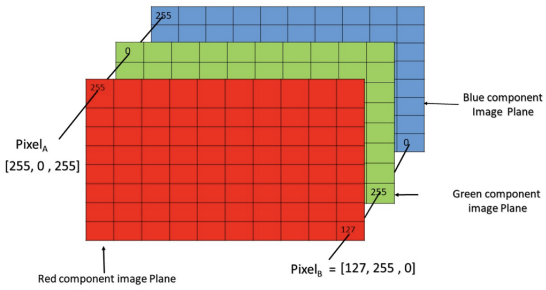

Figure 1: RGB matrix.

We can imagine this as a MxNx3 three-dimensional matrix (RGB matrix), where M and N are the numbers of pixels of the length and width of the image plane, and 3 corresponds to the RGB channels. Each pixel is represented by three integers between 0 and 255 that indicate how intense each color should be for that particular pixel, Fig 12.

This is a fairly simple representation. However, when an image is fed to a Machine Learning model in this format, some issues can occur.

In Figure 2, we can see a cat located on the on the lower left corner of the picture; this can be represented using the three-dimensional matrix described before.

If we move the location of the cat in the picture, the values of three-dimensional matrix change and we get a completely different matrix.

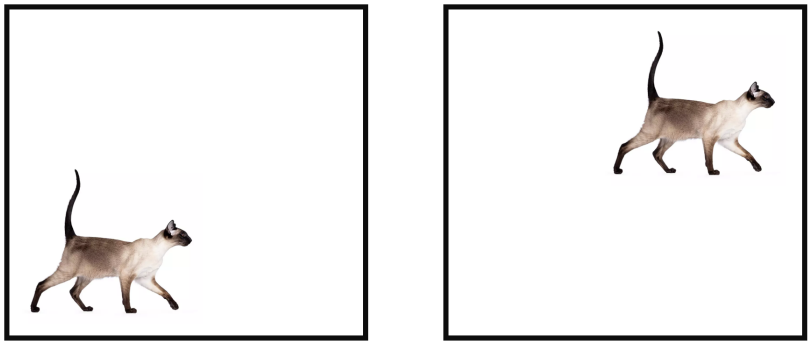

Figure 2: Two images with different underlying RGB matrices.

Suppose we want to classify the picture and identify if there is a cat in it. For humans, it would be natural to recognize that both pictures contains a cat. However, it would be a challenging task for a classification model because the matrices for the two cat images are completely different.

The location of the object in the picture is just one example, but we can think about several more - different intensities of brightness between images, a resized image, different backgrounds, an image rotated by 90 degrees, etc.

In order to solve this issue, other representations were proposed to represent an image.

2 Early Attempt: Bag of Visual Words (BoVW)

Similar to the ”Bag of Words” representation for words, one of the early attempts to solve this problem was to represent the images as a Bag of Visual Words. Recall that Bag of Words uses occurrences of words to describe and extract features for a text. Instead of using actual words, BoVW uses small pieces of the image instead, called patches. Visual features are then extracted from these patches to create a representation of the image.

Features are comprised of Keypoints and Descriptors, where Keypoints are distinctive points within an image that remain constant even if the image is rotated or resized, and the descriptors are used to describe the keypoints.

By utilizing the keypoints and descriptors, we can create vocabularies and generate a frequency histogram of the features found within the image. This histogram can then be used to identify similar images or classify the image into a specific category.

To create a bag of visual words, it is necessary to go through a three-step process, **(1) feature extraction, (2) dictionary/vocabulary construction, and (3) vector quantization.**

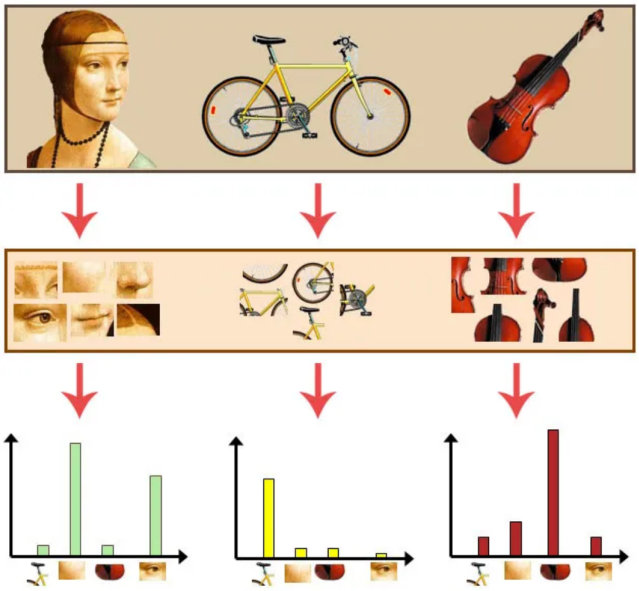

Figure 3: Bag of Visual Words Representation.

2.1 Feature Extraction

This process entails the retrieval of descriptors from every input image. We can accomplish this in many ways. One of the most popular is the SIFT algorithm.

More detail in the SIFT algorithm can be found here.

We can represent the patches as a gray-scaled matrix or as a 2 dimensional matrix that contains the mean RGB values of the pixels. The matrix is then flattened into a vector. The final outputs of this step should be multiple feature vectors.

2.2 Dictionary construction

During the dictionary construction, we go through a clustering process of the feature vectors. For this task, the most popular algorithm used is K-Means, where the centroids of each cluster represents the words in our dictionary of visual words.

A reasonable question to ask here is: ”How many clusters should it have?” This is another difficult question to answer because it can be difficult to find an optimal number. This number will be limited by our computational resources as we have to store detail on each cluster and do some calculation on it in later steps. It is also important to avoid having too many clusters because it could generate an unique cluster for each small variation in our keypoints. This may cause the model to fail to generalize in some cases.

2.3 Vector quantization

After the dictionary is constructed, every image in our data set can be represented as a bag of visual word. In order to do that, we first have to apply feature extraction on the image (Step 1). For each of the feature vectors obtained, we determine the nearest neighbor by calculating the distances between the previously found centroids and the vectors.

Finally, we create a normalized vector of length equal to k, where k is the number of clusters, and the value for each position corresponds to the frequency of the visual word.

This final vector can be used as input to our regression or classification model.

3 Histogram of Oriented Gradients (HOG)

HOG is another technique where features are extracted from oriented gradients within localized portions of an image. The benefit of using gradients is that they provide information about the shape and texture of the objects in an image. For instance, the magnitude of gradients is large around the edges of an object, and small on parts of the image that are smooth or do not contain significant change.

Compared to BoW, this approach therefore uses less information by eliminating factors such as the background or colors of an image and focusing on the gradients to differentiate images from one another.

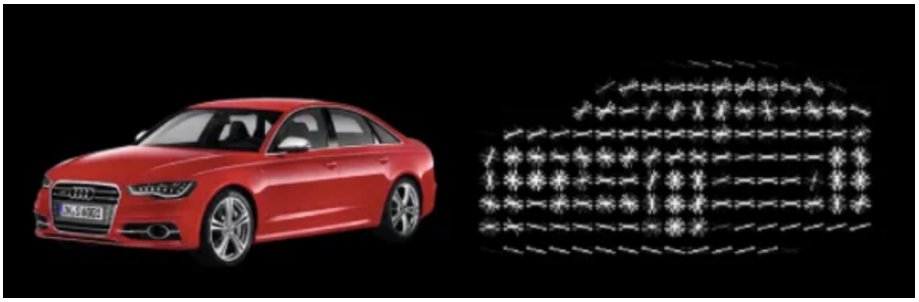

Figure 4: Representation of Oriented Gradients.

The first step to implementing HOG and calculating gradients is to resize the image to a fixed aspect ratio, commonly 128x64 pixels. This ensures that images can be divided into a grid of cells with equal sizes, where gradient orientations can be calculated consistently across images.

Next, gradients are calculated in both x and y directions for each pixel in the cell. Then, the magnitude and the angle of the gradient are derived from these values.

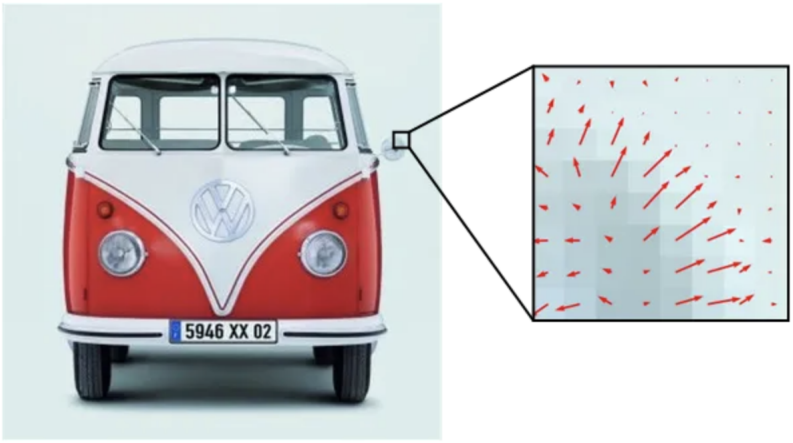

Figure 5: Cell Gradients

Finally, a histogram of gradients is constructed, i.e. the distribution of the gradient orientations per cell. The number of bins is usually set to 9, where each bin width is 20 degrees. Cells are also grouped into overlapping blocks of the same size to improve the accuracy of the feature descriptor. All histograms within a block are added up to form a feature vector which we can eventually feed into our classification network.

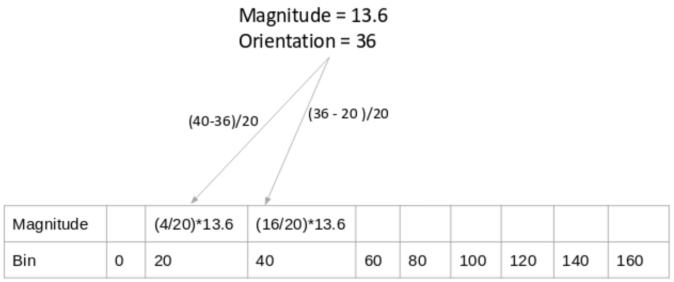

Figure 6: Method of Calculation for B = 9

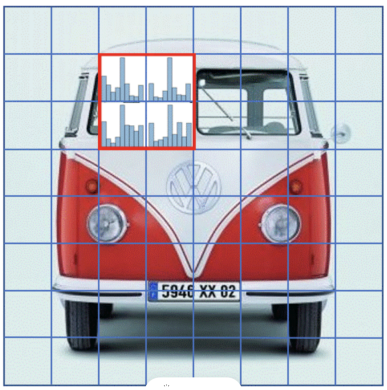

Figure 7: Histogram of Gradients in a Block

4 Convolutional Neural Networks

Convolutional Neural Network is a popular approach in image processing. It imitates how we perceive the environment with our eyes. When we view an image, we subconsciously split it into multiple smaller sub-images and examine them individually. We interpret and analyze the image by combining these sub-images.

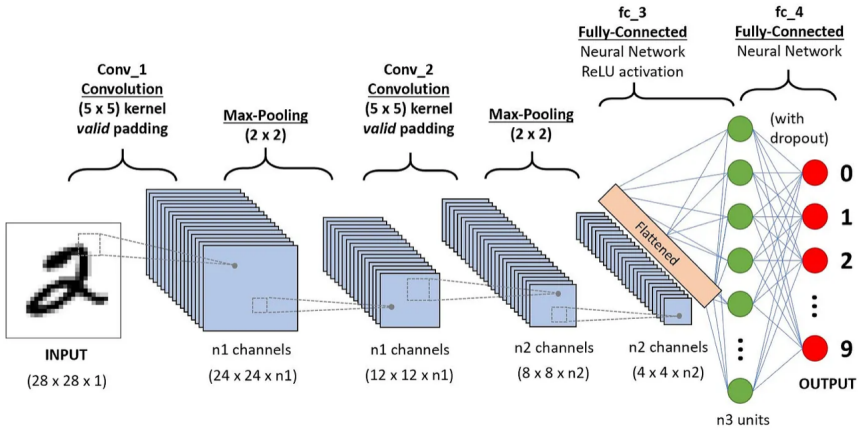

Figure 8: Example of a CNN sequence.

It entails the use of neural networks to do the classification or regression, normally called a fully-connected layer. However, before feeding the neural network, there is a pre-processing on the data that is comprised by two main stages that can be repeated several times, the convolution layer and the pooling layer.

4.1 Convolution layer

In this layer, the high-level features, such as edges, are extracted from the input image. In order to do so, an element called Kernel or Filter is used. If we deal with a RGB channel image, we would have a layer for each channel.

Once we selected our kernels, we apply a convolutional operation on each channel. Suppose we have an image with length and height of 4 and 4 respectively and a kernels of 2x2. The matrix of kernels shifts 9 times and performs an element-wise multiplication operation between the kernel and the portion of image over which the kernel is hovering.

The result for each channel is also summed up element-wise and we obtain a two dimensional matrix, called *Convolved Feature*. We are not limited to only one convolutional layer.

Traditionally, the initial ConvLayer is responsible for identifying the fundamental features of an image, including edges, color, gradient orientation, and so forth. As we incorporate additional layers, the architecture adjusts to detect the higher-level features, leading to a network that has a better comprehension of the images in the dataset.

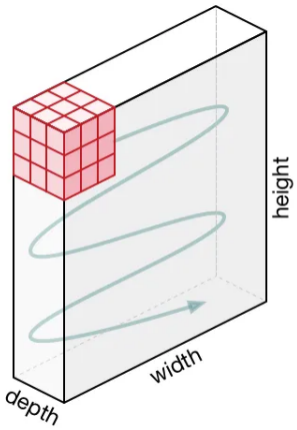

Figure 9: Shifting of the Filter/ Kernel.

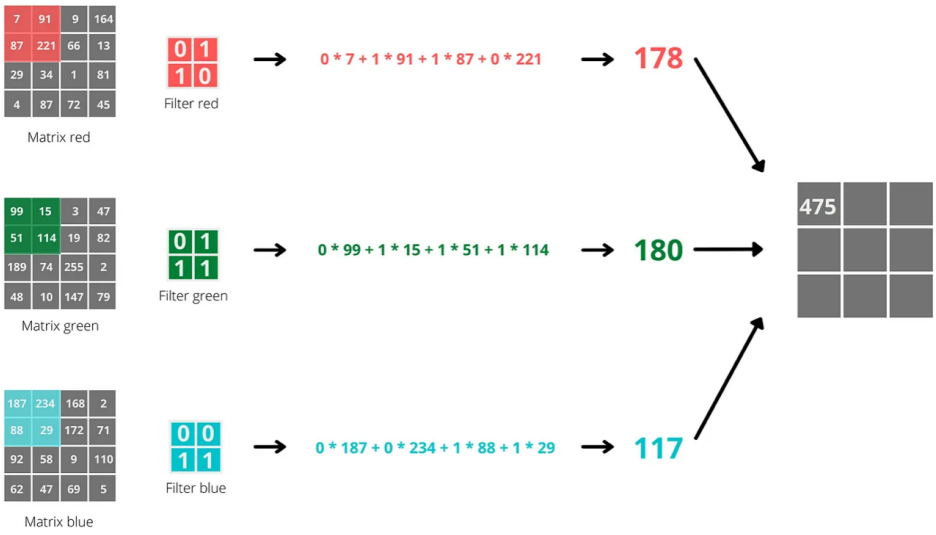

Figure 10: Convolution layer.

The convolved feature matrix may vary in shape depending on the *padding* we chose.

4.1.1 Padding

To get a notion of the padding, we have to understand that we tend to lose information on the pixels location on the perimeter of our image when we apply convolution layers. One solution to this is to add extra filler pixels around the boundary of our image, to increase the effective size of the image.

If we perform the convolutional operation and we end up with a matrix, which shape is the same as the shape of our image, we are referring to a *Same Padding*.

In contrast, if the output matrix's shape is smaller than our initial shape, we are referring to a *Valid Padding*.

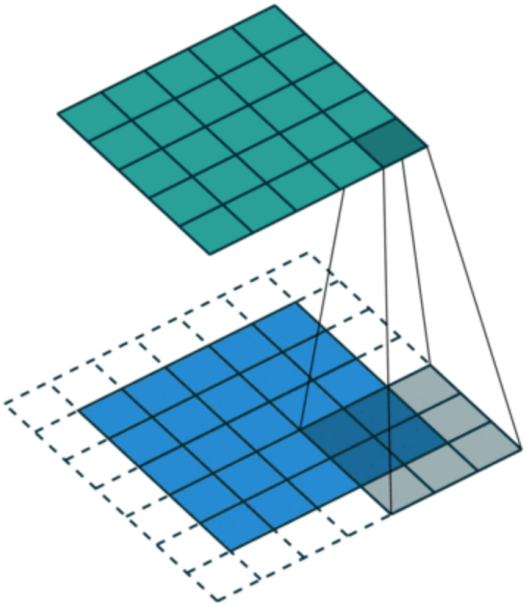

Figure 11: Same Padding visualization.

4.2 Pooling layer

This layer is responsible for reducing the size of our convolved feature, obtained from our convolution layer. The final objective of this layer is to reduce the computational resources needed to handle the data by reducing its dimensionality.

Pooling can be categorized into two types: Max Pooling and Average Pooling. Max Pooling retrieves the highest value within the Kernel-covered segment of the image, while Average Pooling calculates the mean of all the values within the Kernel-covered segment of the image.

Max Pooling has an additional function as a Noise Suppressant, as it eliminates noisy activations. In contrast, Average Pooling functions only perform dimensionality reduction. Therefore, most of the times Max Pooling outperforms Average Pooling significantly.

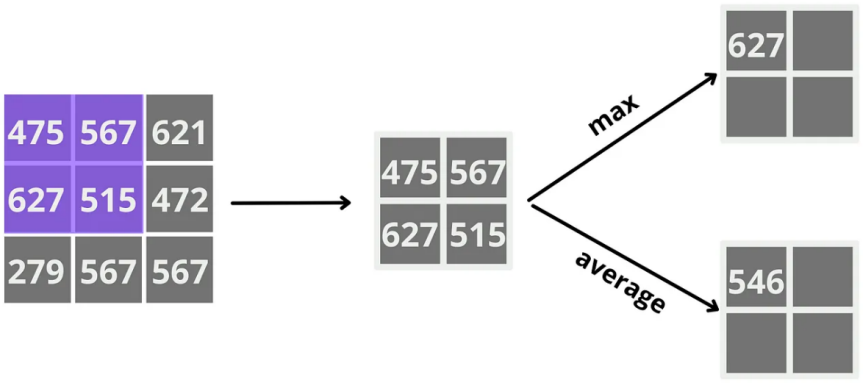

Figure 12: Pooling layer.

To capture low-level details in images, the number of layers may need to be increased depending on their complexity. However, this comes at the expense of increased computational power.

Following the convultion and pooling process, the model is now capable of understanding the features. The next step involves flattening the final output and feeding it into a standard Neural Network for classification purposes (Fully-Connected Layer).

5 References

- https://towardsdatascience.com/using-convolutional-neural-network-for-image-classification-5997bfd0ede4
- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f
- https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa
- https://customers.pyimagesearch.com/the-bag-of-visual-words-model/
- https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40
- https://www.pinecone.io/learn/bag-of-visual-words/
- https://www.geeksforgeeks.org/matlab-rgb-image-representation/