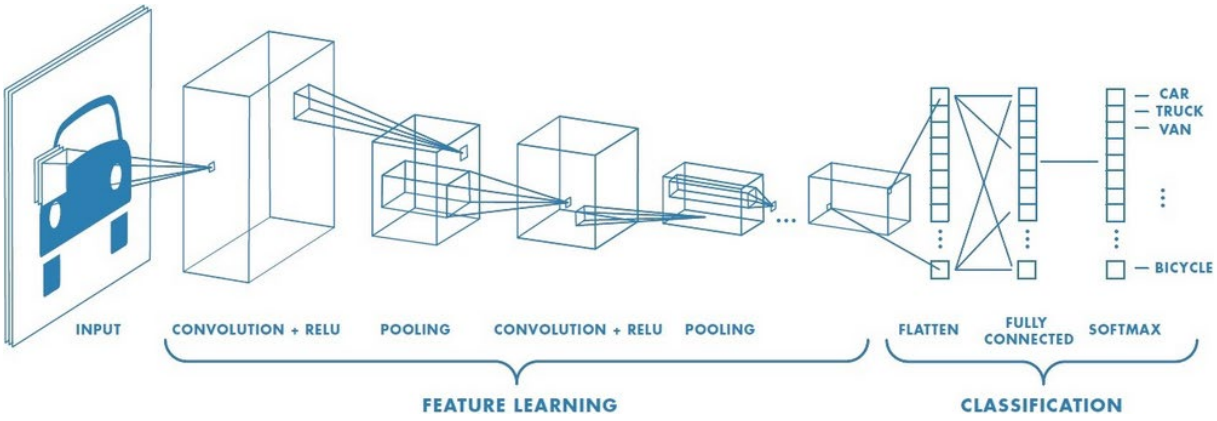
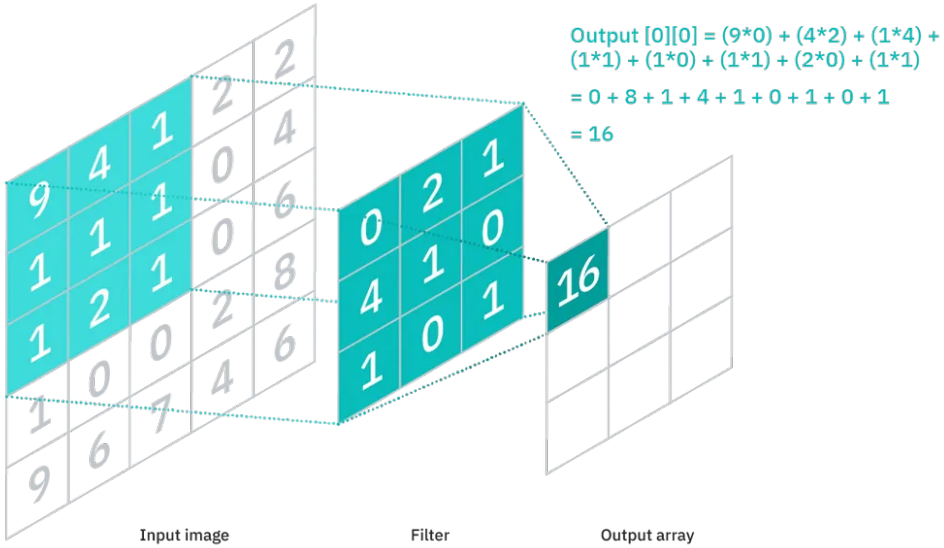
This week, we worked on learning - reinforcing a general functioning for the CNN model mentioned in the Proposal Report. In the code we worked with, we designed the CNN model from scratch using PyTorch and tested it on the simple Cifar10 dataset. We want to emphasize that no proposals were made and that we spent the week just understanding how the CNN model works in detail.We will be building Convolutional Neural Networks (CNNs) from scratch in PyTorch, and seeing them in action as we train and test them on a real-world dataset.

We will start by exploring what CNNs are and how they work. We will then look into PyTorch and start by loading the CIFAR10 dataset using torchvision (a library containing various datasets and helper functions related to computer vision). We will then build and train our CNN from scratch. Finally, we will test our model.

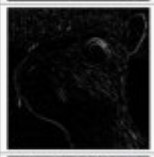
A convolutional neural network (CNN) takes an input image and classifies it into any of the output classes. Each image passes through a series of different layers – primarily convolutional layers, pooling layers, and fully connected layers. The below picture summarizes what an image passes through in a CNN:



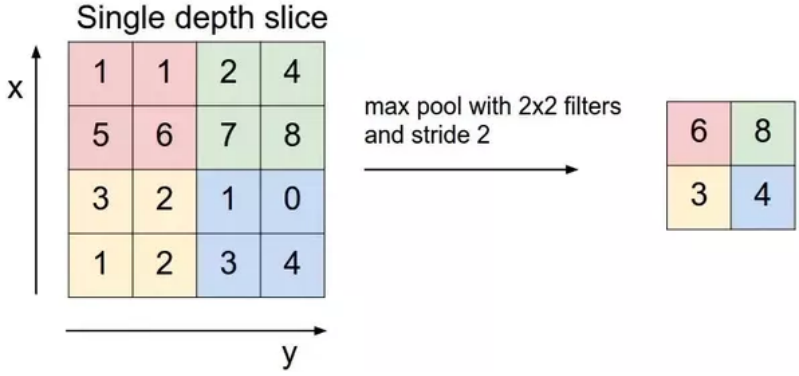
The convolutional layer is used to extract features from the input image. It is a mathematical operation between the input image and the kernel (filter). The filter is passed through the image and the output is calculated as follows:



Different filters are used to extract different kinds of features. Some common features are given the following image:

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Pooling layers are used to reduce the size of any image while maintaining the most important features. The most common types of pooling layers used are max and average pooling which take the max and the average value respectively from the given size of the filter (i.e, 2x2, 3x3, and so on).Max pooling, for example, would work as follows:



PyTorch is one of the most popular and widely used deep learning libraries – especially within academic research. It's an open-source machine learning framework that accelerates the path from research prototyping to production deployment and we'll be using it today in this article to create our first CNN. We will be using the CIFAR-10 dataset. The dataset has 60,000 color images (RGB) at 32px x 32px belonging to 10 different classes (6000 images/class). The dataset is divided into 50,000 training and 10,000 testing images.

The steps of loading the dataset and designing the model are described in more detail in the notebook with the code. The loss values and accuracy values of the model at each epoch are given below.

```
Epoch [1/20], Loss: 1.8847
Epoch [2/20], Loss: 1.5313
Epoch [3/20], Loss: 1.7422
Epoch [4/20], Loss: 1.3382
Epoch [5/20], Loss: 1.9132
Epoch [6/20], Loss: 1.0077
Epoch [7/20], Loss: 1.3437
Epoch [8/20], Loss: 0.4538
Epoch [9/20], Loss: 1.4301
Epoch [10/20], Loss: 1.1297
Epoch [11/20], Loss: 1.1042
Epoch [12/20], Loss: 0.6595
Epoch [13/20], Loss: 0.8148
Epoch [14/20], Loss: 0.7682
Epoch [15/20], Loss: 0.8232
Epoch [16/20], Loss: 0.8483
Epoch [17/20], Loss: 0.9479
Epoch [18/20], Loss: 0.8178
Epoch [19/20], Loss: 0.3819
Epoch [20/20], Loss: 0.6994
```

```
Accuracy of the network on the 50000 train images: 83.356 %
```

Here is the link to the notebook we are working on to learn the CNN model and consolidate our knowledge: https://github.com/ogulcanakcaa/Fashion-Recommendation-System/blob/main/Codes/Writing_CNNs_from_Scratch_in_PyTorch.ipynb

Tahir Kurtar

Oğulcan Akca