18 - Prática: Visão Computacional (III)

Lucas Samuel Zanchet

Tensors are a fundamental data structure from mathematics to develop Deep Learning models. In the context of Data Science they are multi-dimensional arrays of numbers that represent complex data.

The advantage of using tensors from PyTorch instead of Numpy arrays is that PyTorch makes use of the GPU for parallel computing.

An activation function is a function that changes the output of a neuron. Some examples are the hyperbolic tangent, sigmoid, rectified linear unit, etc.. Their purpose is to enable non-linearity in the model, since the nature of weights and biases only allow for linear mappings.

A loss function is a calculation performed at the end of a batch run which measures the quality of the model. A loss of 0 would be a perfect model. The loss function is also what drives the gradient descent.

The optimizer is a algorithm whose goal is to lower the loss of a network. It does it by changing the weights and biases to make the loss go to a point of minimum. Since a model is basically a function, the most basic optimizer algorithm is the gradient descent, a long time known numerical algorithm to find local minima in functions. Other algorithms such as RMSprop, Adam, Adagrad are variations of the gradient descent which implement other techniques like momentum to optimize the convergence.

PyTorch gives us a tool called DataLoader that makes it easy to load and set data for training more easily and efficiently. It efficiently manages datasets, handles shuffling, and provides parallel data loading. Developers can focus on model design and leave data logistics to DataLoader.

Tuning the hyperparameters is a task of the developer. They are the settings that will dictate the structure of the network and how it will learn. Tuning them involves experimentation and validation. The right hyperparameters lead to better generalization and faster convergence.

Normalization of images is an important step in preprocessing the data to train a computer vision model. Normalization ensures consistent input across different images. Common techniques include scaling pixel values to [0, 1] or standardizing them (zero mean, unit variance).

When dealing with images we use convolution and pooling layers to detect features in the image. Convolutional layers detect local features (edges, textures) in images. Pooling layers downsample feature maps, reducing spatial dimensions. Convolutional Neural Networks (CNNs) leverage these layers to extract hierarchical representations from images. They work by applying filters, or also called kernels, to an image, which in training converge to detect a specific feature or pattern.

Image augmentation is a technique used to battle overfitting in a restricted dataset. The idea is to modify the image so that the CNN learns without any bias related to position, rotation, scale, birghtness, etc.. Unlike Tensorflow, PyTorch doesn't come with a data augmentation library included, so we use the imgaug library.

An Autoencoder Neural Network is a modern type of NN that is able to encode and decode an image. It is used to compress data and for object classification. The encoder compresses the image to a lower dimensionality in a latent space. The decoder generates back the image using just the information from the latent space.

Variational Autoencoders are a variation of autoencoders that use bayesian inference. They can be user not only to compress data, but to generate new images by sampling from the learned data. They are the base of modern NN like ChatGPT and Image generation models.

In the last hands on example about image colorization the professor uses a UNet architecture. This architecture was not covered in the course, but through some research I found that it is used typically in image classification and object detection tasks.