. Intro to CNNs

Let's introduce Convolutional Neural Networks, a different type of network that has led to a lot of advances in computer vision, as well as in many other areas.

2. How do they work?

A convolutional model uses convolutional layers. A convolution is a simple mathematical operation that preserves spatial relationships. When applied to images it can detect relevant areas of interest like edges, corners, vertical lines, etc.

3. Convolutions demonstration

It consists of applying a filter, also known as kernel, of a given size. In this image, we are applying a 3 by 3 kernel. We center the kernel matrix of numbers as we slide through each pixel in the image, multiplying the kernel and pixel values at each location and averaging the sum of values obtained. This effectively computes a new image where certain characteristics are amplified depending on the filter used. The secret sauce of CNNs resides in letting the network itself find the best filter values and to combine them to achieve a given task.

4. Typical architectures

For a classification problem with many possible classes, CNNs tend to become very deep. Architectures consist of concatenations of convolutional layers among other layers known as pooling layers, that we won't cover here. Convolutional layers perform feature learning, we then flatten the outputs into a unidimensional vector and pass it to fully connected layers that carry out classification.

5. Input shape to convolutional neural networks

Images are 3D tensors, they have width, height, and depth. This depth is given by the color channels. If we use black and white images we will just have one channel, so the depth will be 1.

6. How to build a simple convolutional net in keras?

To build a CNN in Keras we first import the Conv2D and Flatten layers from tensorflow.keras.layers. We instantiate our model and add a convolutional layer. This first convolutional layer has 32 filters, this means it will learn 32 different convolutional masks. These masks will be squares of 3 by 3 as defined in the kernel_size. For 28 times 28 black and white images with only one channel, we use an input shape of (28, 28, 1). We can use any activation, as usual. We then add another convolutional layer and end flattening this 2D layer into a unidimensional layer with the Flatten layer. We finish with an output dense layer.

7. Deep convolutional models

ResNet50 is a 50 layer-deep model that performs well on the Imagenet Dataset, a huge dataset of more than 14 million images. ResNet50 can distinguish between 1000 different classes. This model would take too long to train on a regular computer, but Keras makes it easy for us to use it. We just need to prepare the image we want to classify for the model, predict the processed image, and decode the predictions!

8. Pre-processing images for ResNet50

To use pre-trained models to classify images, we first have to adapt these images so that they can be understood by the model. To prepare images for ResNet50 we would do the following. First import the image from tensorflow.keras.preprocessing and preprocess_input from tensorflow.keras.applications.resnet50. We then load our image with load_img, providing the target size, for this particular model that is 224 by 224. We turn the image into a numpy array with img_to_array, we expand the dimensions of the array and preprocess the input in the same way the training images were.

9. Using the ResNet50 model in Keras

We import ResNet50 and decode_predictions,load the model with Imagenet pre_trained weights,predict on our image,and decode the predictions. That is, getting the predicted classes with the highest probabilities.

10. What is going on inside a convnet?

Inside a CNN we can check how the different filters activate in response to an input image. We will explore this in the exercises!

11. Let's experiment!

Let's experiment with convolutional networks!