University of São Paulo
Institute of Mathematics and Statistics
Bachelor of Computer Science

A Survey of Explainable AI (XAI) Methods for Convolutional Neural Networks

Antonio Fernando Silva e Cruz Filho
João Gabriel Andrade de Araujo Josephik

Final Essay

mac 499 — Capstone Project

Supervisor: Prof. Dr. Nina S. T. Hirata

São Paulo
2024

Agradecimentos

Do. Or do not. There is no try.
— Mestre Yoda

Texto texto. Texto opcional.

Resumo

Antonio Fernando Silva e Cruz Filho
João Gabriel Andrade de Araujo Josephik. **Title of the document: *a subtitle***. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2024.

Elemento obrigatório, constituído de uma sequência de frases concisas e objetivas, em forma de texto. Deve apresentar os objetivos, métodos empregados, resultados e conclusões. O resumo deve ser redigido em parágrafo único, conter no máximo 500 palavras e ser seguido dos termos representativos do conteúdo do trabalho (palavras-chave). Deve ser precedido da referência do documento. Texto texto. Texto texto.

Palavras-chave: Palavra-chave1. Palavra-chave2. Palavra-chave3.

Abstract

Antonio Fernando Silva e Cruz Filho
João Gabriel Andrade de Araujo Josephik. **A Survey of Explainable AI (XAI) Methods for Convolutional Neural Networks**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2024.

Elemento obrigatório, elaborado com as mesmas características do resumo em língua portuguesa. De acordo com o Regimento da Pós-Graduação da USP (Artigo 99), deve ser redigido em inglês para fins de divulgação. É uma boa ideia usar o sítio www.grammarly.com na preparação de textos em inglês. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text. Text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text text.

Keywords: Keyword1. Keyword2. Keyword3.

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
XAI	Explainable AI
MLP	Multilayer Perceptron
CNN	Convolutional Neural Network
Conv	Convolution
IME	Institute of Mathematics and Statistics
USP	University of São Paulo

List of Figures

List of Tables

List of Programs

Contents

Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Other Section Example

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Next Section example

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Chapter 1

Background

In this chapter, we will introduce important concepts required for the understanding of this study. We begin by introducing Explainable AI (XAI) and its principles. We also introduce Neural Networks and important concepts such as Gradient Descent and Back Propagation. Finally, we introduce Convolutional Neural Networks, the main focus of this study.

1.1 Explainable AI

With the rise of Machine Learning models in the last decade in the business and academic areas, Artificial Intelligence (AI) is becoming increasingly present in important decision-making tasks. However, as AI models have become more sophisticated, particularly with the advent of Deep Learning techniques, their internal workings have often remained opaque. Explainable AI (XAI) aims to make models and their decisions more transparent, interpretable and understandable to both experts and inexperienced users.

1.1.1 What is Explainable AI?

Defining a mathematical formalization to explainability of Machine Learning is a difficult task considering the subjective nature of what one may consider "explainable". In non-mathematical terms, Explainability in AI refers to the capacity to articulate or justify the behavior of a model, focusing on methods that explain a model's decisions after they are made.

Another important concept in the area is Interpretability, which can be defined as "the degree to which a human can understand the cause of a decision" by Miller (2017).[1] In this case, however, a model's decision is understandable entirely by its inherent transparency. In other terms, the model is simple enough to be interpretable by a human directly, without the use of external techniques.

[1] Miller, Tim. "Explanation in artificial intelligence: Insights from the social sciences." arXiv Preprint arXiv:1706.07269. (2017).

Models with low complexity whose decisions are understandable by humans are defined as *Interpretable Models*. Linear Regression, Logistic Regression and Decision Tree models are examples of models classified as *Interpretable Models*. Now, models with a level of complexity that prevents humans from directly understanding their decision-making processes are referred to as *Explainable Models*. Recently popular *Deep Learning Models* are one kind of *Explainable Models* and will be the main focus of this essay, especially *Deep Convolutional Neural Networks*, explored in section 1.3.

1.1.2 Why Explainable AI is Necessary

Creating explanations to a model's decisions can yield many advantages, including increase in model trust, more ethical and fair decisions, correctly following regulatory compliances and easier model debugging.

While Debugging, Machine Learning models can have quite unpredictable behavior, detecting biases not initially noticed by humans. This can yield great performance in the training or even validation and test datasets, but poor performance in real world deployment. For example, while training an image classifier on dog and cat images one can find great accuracy on classification of dogs over green fields. However, when analyzing what regions of the image a model "sees" for the prediction, researchers may find out that the model is actually looking at the image background, since dog owners tend to take their pets for walks more often than cat owners, therefore making dog pictures with a green background more likely than pictures with cats over green fields. Visualizing regions of images that our vision model "sees" is only possible with Explainable AI techniques, such as GradCAM (Selvaraju *et al.*, 2019) and other Gradient Saliency methods.

1.2 Gradient Descent

Let $f : A \to B$ where $A \subseteq \mathbb{R}^n$ for $n \in \mathbb{N}$ and $B \subseteq \mathbb{R}^+$. Suppose we want to find the solution to the optimization problem

$$\underset{x \in A}{\operatorname{argmin}} f(x) \tag{1.1}$$

when $\frac{\partial f}{\partial x}$ is known for any value of x. Considering that the vector $\frac{\partial f}{\partial x}$ points to the direction of the steepest ascent of the function, the vector $-\frac{\partial f}{\partial x}$ will point to the steepest descent from the given point x. Therefore, one can define an initial random value for x and update x using $-\frac{\partial f}{\partial x}$ and a scaling factor η in order to find a local minimum of f and an approximation to the solution of given optimization problem.

We can define such method using the following formula, where x_t represents the value of x at iteration t of the algorithm:

$$x_{t+1} = x_t - \eta \frac{\partial f}{\partial x}. \tag{1.2}$$

The term η is often called the *learning rate* used in the Gradient Descent method and is often defined manually by the user.

The Gradient Descent method can be used to optimize a neural network's parameters to solve a given problem using a *loss function*.

1.3 Neural Networks

Neural Networks are proven to be universal approximators.[1] That means that Neural Networks are Machine Learning models capable of representing any continuous function, therefore making Neural networks adept at modeling a range of different complex problems. This class of models have seen a growing presence across both academic and industry landscapes. However, given the architecture of multiple hidden layers of Neural Networks creating complex internal patterns, such models are classified as Explainable Models.

In this section, the inner workings of Neural Networks will be explained, starting with the *Perceptron*, considered the fundamental building block of Neural Networks.

1.3.1 Perceptron

A Perceptron is a Machine Learning model inspired by how biological neurons work. It is a simple binary linear classifier that defines its parameters by linear combinations of points in the dataset. The Perceptron model can be described by Figure 1.1.

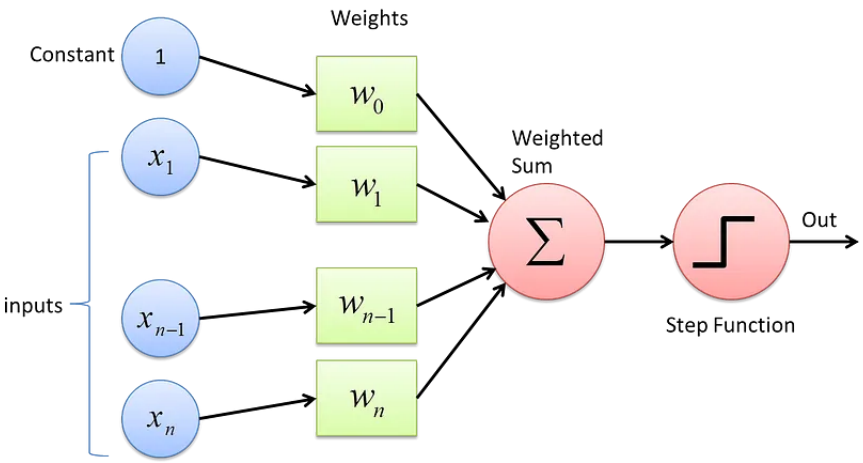


Figure 1.1: *Perceptron Architecture. Font:* Towards Data Science[2].

Where the weights w_i for $i \in \{0, 1, \cdots, n\}$ are trainable parameters and the step function can be defined as $\sigma : \mathbb{R} \to \{0, 1\}$ such that

[1] Hornik, K., Stinchcombe, M., White, H. Multilayer feedforward networks are universal approximators. https://doi.org/10.1016/0893-6080(89)90020-8

[2] https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

$$\sigma(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases} \tag{1.3}$$

Therefore, the Perceptron model can be defined as the function $f : \mathbb{R}^n \to \{0, 1\}$ where

$$f(x) = \sigma(w_0 + \sum_{i=1}^{n} w_i x_i). \tag{1.4}$$

The Perceptron model updates its parameters using each sample (x, y) of the dataset with the rule

$$w_i^{t+1} = w_i^t + \eta\,(y - f(x))x_i \tag{1.5}$$

for $i \in \{1, \cdots, n\}$ and

$$w_0^{t+1} = w_0^t + \eta\,(y - f(x)), \tag{1.6}$$

where η is the *learning rate* hyperparameter and t is the update iteration number.

As a linear model, the Perceptron can only model linear problems, which only represent a small subset of real world problems. As a solution, researchers started combining Perceptrons in a layered structure. The multiple layers consisting of multiple Perceptrons created a structure capable of representing more complex functions, or more specifically, *any continuous function*. Such architecture is called a *Multilayer Perceptron* (MLP). Neural Networks are considered *"softened" MLPs*,[3] which means that those models use differentiable activation functions that allow for gradient-based optimization, using algorithms like the *Gradient Descent*.

1.3.2 How Neural Networks Work

Neural Networks work by using multiple Perceptrons with differentiable activation functions grouped in a layered structure in order to model complex functions. Such model's architecture can be described by Figure 1.2 in a network similar to a graph.

[3] Terminology used by Professor Yaser Abu-Mostafa, in the famous Machine Learning Book *Learning from Data*. Source: https://work.caltech.edu/telecourse

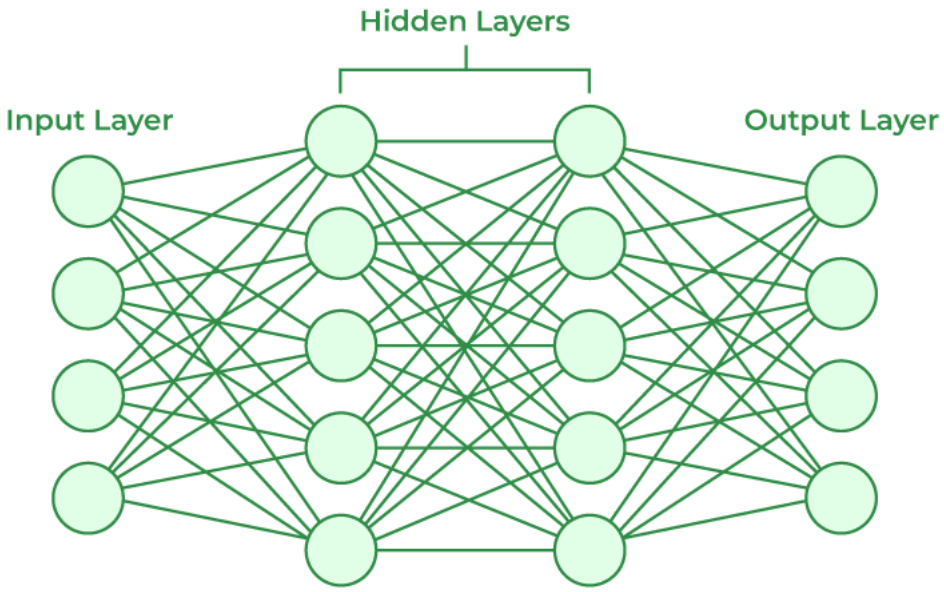


Figure 1.2: *Neural Network Architecture. Font:* Geeks For Geeks[4].

Each node of such graph represents a *neuron*, which is a single Perceptron model with each edge representing an output of a neuron being passed to another neuron as input. Each edge will have a weight associated to it, representing the importance of a neuron from a previous layer. Inference in this model is called *Forward Propagation*, where the input is passed in the *Input Layer* and is computed by calculating a Perceptron's output and passing it to the next layer, until the *Output Layer* is reached. Layers in between are called *Hidden Layers*.

Mathematically, a neuron can be described by the same formula in equation 1.4, but with a different activation function.

1.4 Convolutional Neural Networks

1.4.1 Convolutions

First, it is important to define what a convolution is. Given two discrete one-dimensional signals f and g, their convolution $f * g$ is defined as:

$$(f * g)[n] = \sum_{i=-\infty}^{+\infty} f[n]g[n-i]$$

Given two discrete two-dimensional signals f and g, their convolution $f * g$ is calculated as:

[4] https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

$$(f * g)[m][n] = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f[m][n] \cdot g[m-i][n-j]$$

In practice, the signal g is represented by a window (or kernel), usually square and of odd size. Thus, we can abstract convolution as the multiplication of a sliding window. The picture below illustrates this process. It is important to note that the window needs to be flipped during the convolution, although this is not illustrated in the picture.

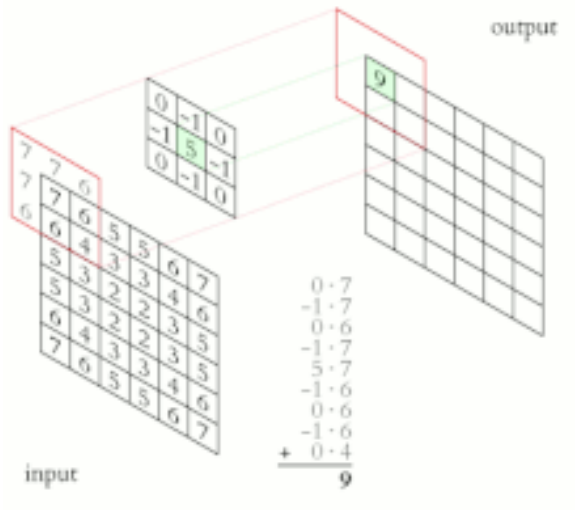


Figure 1.3: *2D Convolution*

This process can be used to apply different filters to images. For example: using a 3×3 window with all weights equal to $\frac{1}{9}$, we can generate a filter that blurs the image (moving average). Below is an example of applying this filter:

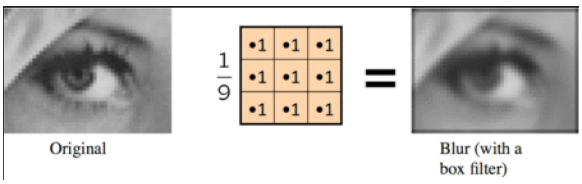


Figure 1.4: *Filtered Image (3x3 Mean Filter)*

It is also important to note that convolution is commonly implemented in machine learning contexts as "cross-correlation," which is a very similar operation but without the flipping of the window. Note that, since the weights are learned in our case, there is no difference. Therefore, in our context, convolution and cross-correlation are synonymous.

A pertinent question that can be asked is what happens at the edges of the image. When the window is sliding over them, what happens to the missing pixels? The process of filling in these pixels is called padding. Padding can be done with zeros, the nearest pixel, or not be done at all. Note that when there is no padding, the image decreases in size after convolution.

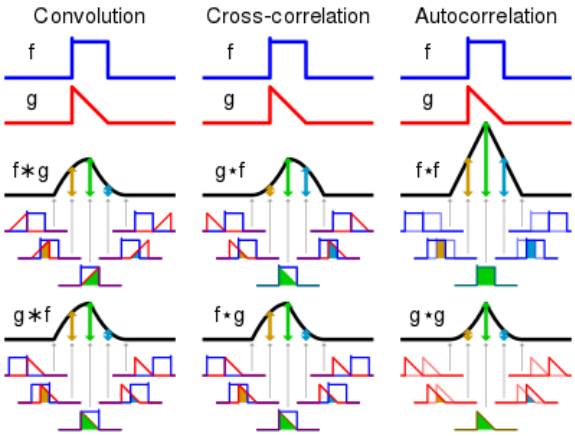


Figure 1.5: *Comparison with Cross-Correlation*

Another important hyperparameter that can be adjusted is the "stride." This defines how many positions the window is moved at a time. That is: a stride value different from 1 also implies a decrease in image size after convolution.

1.4.2 Convolutional Layer

In order to understand the need for convolutional networks, we have to understand why it's impractical to use fully-connected networks to process images. Let's walk though an example to see that.

Consider a color image with dimensions 512x512. If we were to process this image with a conventional neural network, the input layer would have $3 \cdot 512 \cdot 512 = 786432$ dimensions. Assume a hidden layer with only 128 neurons (which is relatively small). Just between these two layers, there would be 100663296 parameters! This is highly inefficient.

The solution to this problem is to extract features from the image, which will serve as input to the network. These features could include various aspects such as symmetry, black levels, contrast, presence or absence of patterns, etc. All of these features will serve as input to the network. As a result, we can reduce the input layer's dimensionality from several hundred thousand to just a few dozen.

However, a challenge still remains: how do we select these features? We can apply convolutions to the image to calculate interesting features, and these convolutions can be learned alongside the rest of the network! It is important to understand some essential details about these networks before proceeding. Each convolutional layer has three dimensions: height, width, and the number of channels. The input layer typically has one channel for black and white images, or three channels for color images.

Each channel in each convolutional layer combines all the channels from the previous layer. In other words, the "windows" used have weights for all the channels. These windows slide over the data from the previous layer to generate **one channel** in the next layer.

Let us consider an example. Suppose we have a network that processes color images of size 128×128 pixels. This network has 3 convolutional layers with 16, 32, and 64 channels per layer, respectively. Assume a window size of 3 for all layers. In this case, we have:

- First layer: window size is $3 \times 3 \times 3$. With 16 output channels, we will have

$$16 \cdot 3 \cdot 3 \cdot 3 = 432 \text{ parameters.}$$

- Second layer: window size is $3 \times 3 \times 16$. With 32 output channels, we will have

$$32 \cdot 3 \cdot 3 \cdot 16 = 4608 \text{ parameters.}$$

- Third layer: window size is $3 \times 3 \times 32$. With 64 output channels, we will have

$$64 \cdot 3 \cdot 3 \cdot 32 = 18432 \text{ parameters.}$$

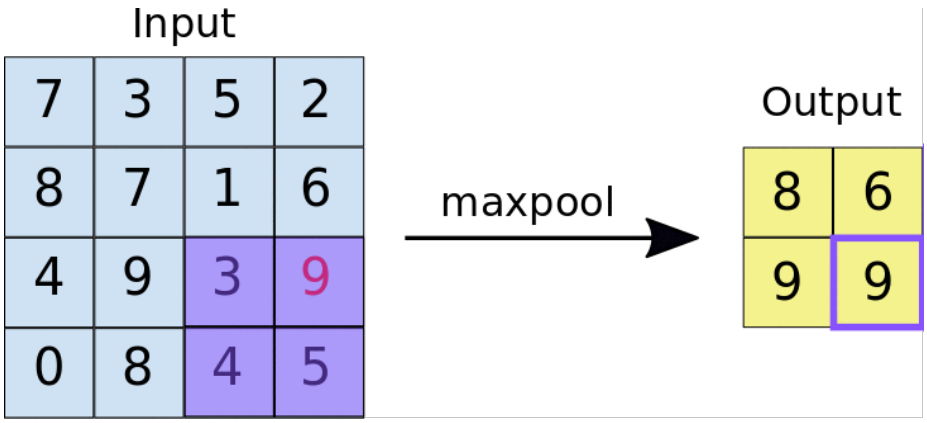
Another important detail is the output dimension of each layer. This depends on whether or not **padding** is used. **Padding** refers to how the layer behaves at the image edges. We can complete the image with zeros, the nearest pixel value, or the pixel value from the opposite edge of the image. If padding is not used, the output dimensions will decrease by $2\lfloor \frac{W}{2} \rfloor$, where W is the window size. For instance, with a window size of 3, each layer will reduce the image size by 2 pixels. If the input is 128×128, the output of the first layer will be 126×126, the second layer will output 124×124, and so on.

1.4.3 Pooling

Remember, each convolution extracts a feature from the image. Therefore, when we perform another convolution using the outputs from the previous layer, we are combining features extracted from the image to compute new features. As a result, deeper layers extract more complex features from the image. For instance, the first layer may extract features like the presence of vertical straight lines, while the tenth layer may extract features like "presence of dog snouts."

Thus, the features involved gradually become less localized and more global (pertaining to the entire image). This is why it is useful to summarize information into smaller dimensions as the network deepens.

To accomplish this, we use "pooling" layers. These layers work similarly to convolutions: windows slide over the data and compute an output based on nearby pixels. However, this time, a function is used to aggregate these data. Common functions include "max" (maximum value) and "avg" (average value).

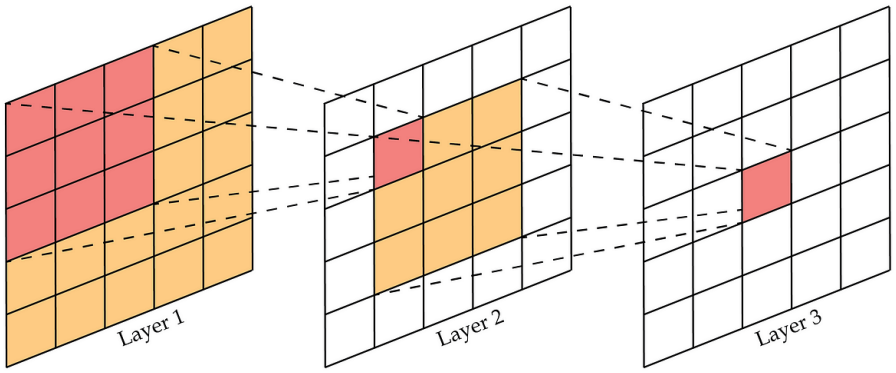


1.4.4 Receptive Field

An important concept for understanding the power of **deep convolutional networks** is the **receptive field**. This concept relates to the power that chained convolutions have.

Consider an input image. Apply a 3×3 convolution to it. Now, apply another 3×3 convolution to the output of the first convolution. Observe this output image. How much information does each pixel contain about its neighbors?

Receptive Field in Convolutional Networks



The answer is that each pixel contains information from a region of size 5×5 around it! This is the receptive field of these neurons.

A common misconception is that, since the receptive field of two 3×3 convolutions is 5×5, two 3×3 convolutions have the same **expressive power** as a 5×5 convolution. This **is not** true. A 5×5 convolution has 25 parameters, while two chained 3×3 convolutions only have 18 parameters.

References

[Selvaraju *et al.* 2019] Ramprasaath R. Selvaraju *et al.* "Grad-cam: visual explanations from deep networks via gradient-based localization". *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7 (cit. on p. 4).

Index