Sensores espectroscópicos e modelos de regressão aplicados na análise de solos

Aula 4 – Redes Neurais Convolucionais

Me. José Vinícius Ribeiro

Universidade Estadual de Londrina

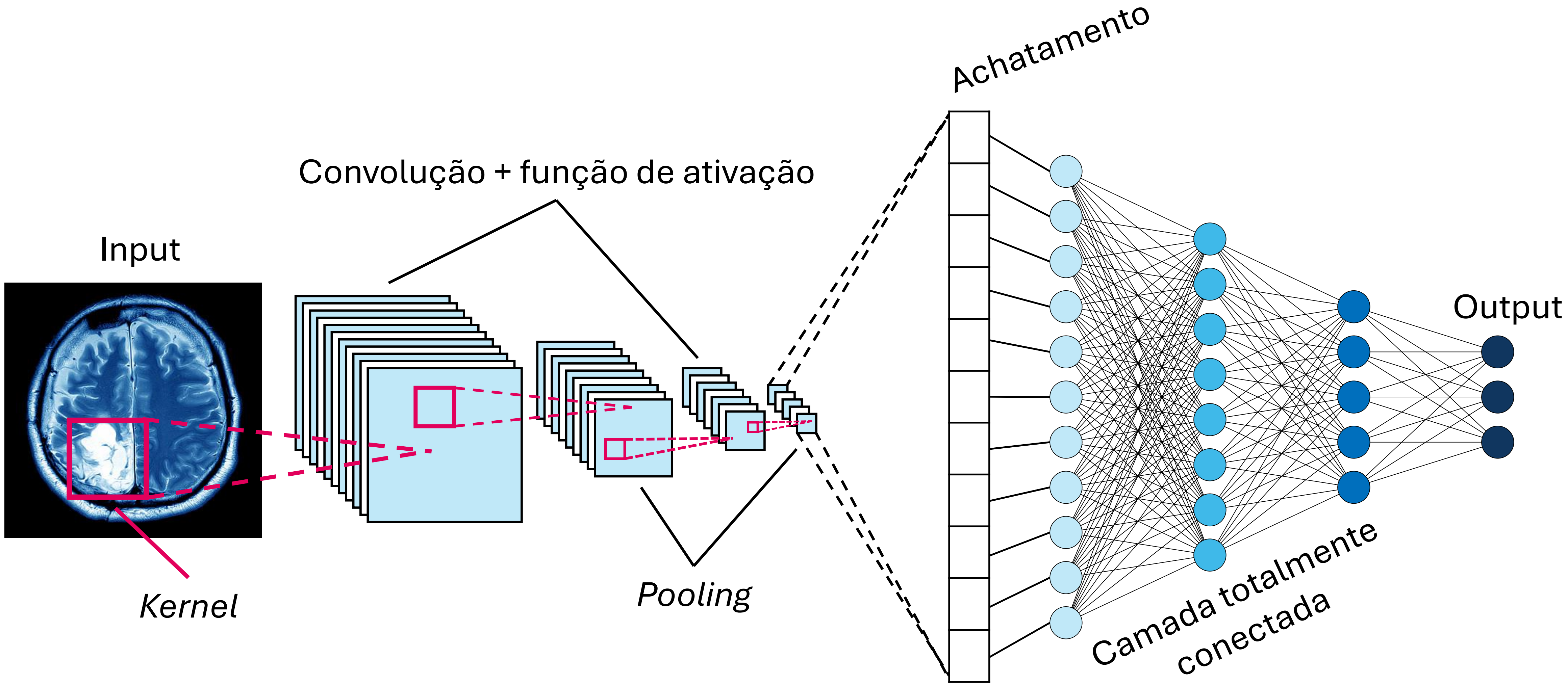
PÓS GRADUAÇÃO FÍSICA UEL

SUMÁRIO

- Motivação
- Camada de convolução
- Camada de *pooling*
- Visualizando os *features maps*
- CNN 1D – lidando com espectros
- CNN 2D – lidando com espectrogramas
- Noções de *transfer learning*
- Prática no python (vscode)

Arquitetura básica

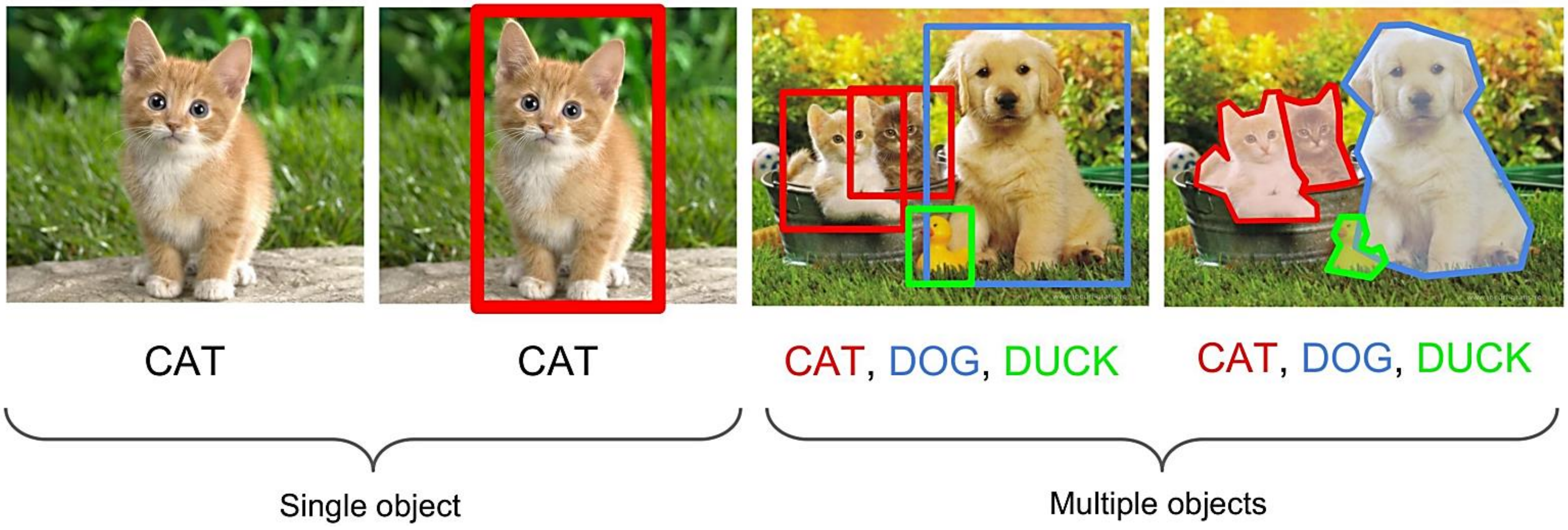
Redes Neurais Convolucionais (CNN)



MOTIVAÇÃO

CNN - motivação

Redes Neurais Convolucionais (CNNs) são algoritmos de IA baseados em redes neurais multicamadas que aprendem características relevantes a partir de imagens, sendo capazes de realizar diversas tarefas como classificação, detecção e segmentação de objetos

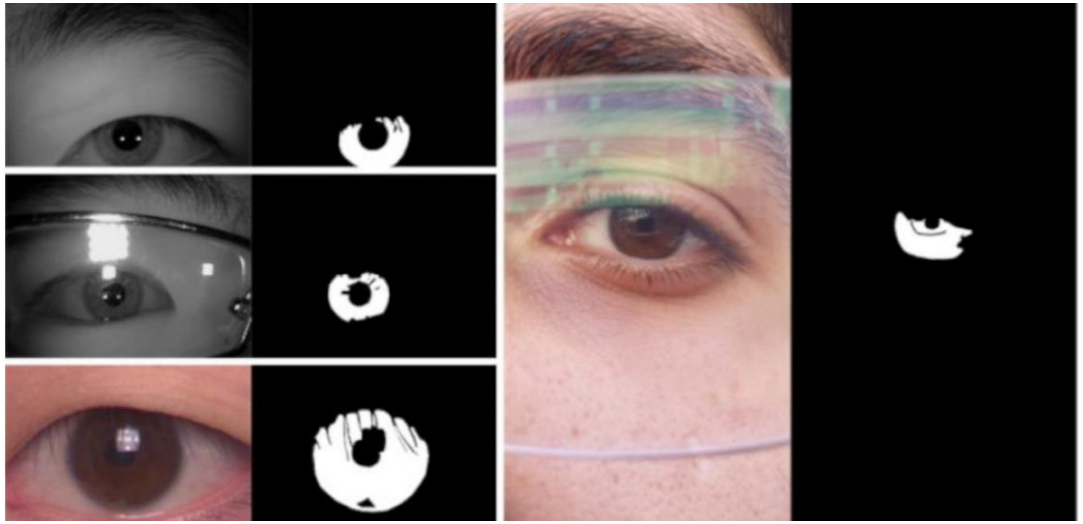


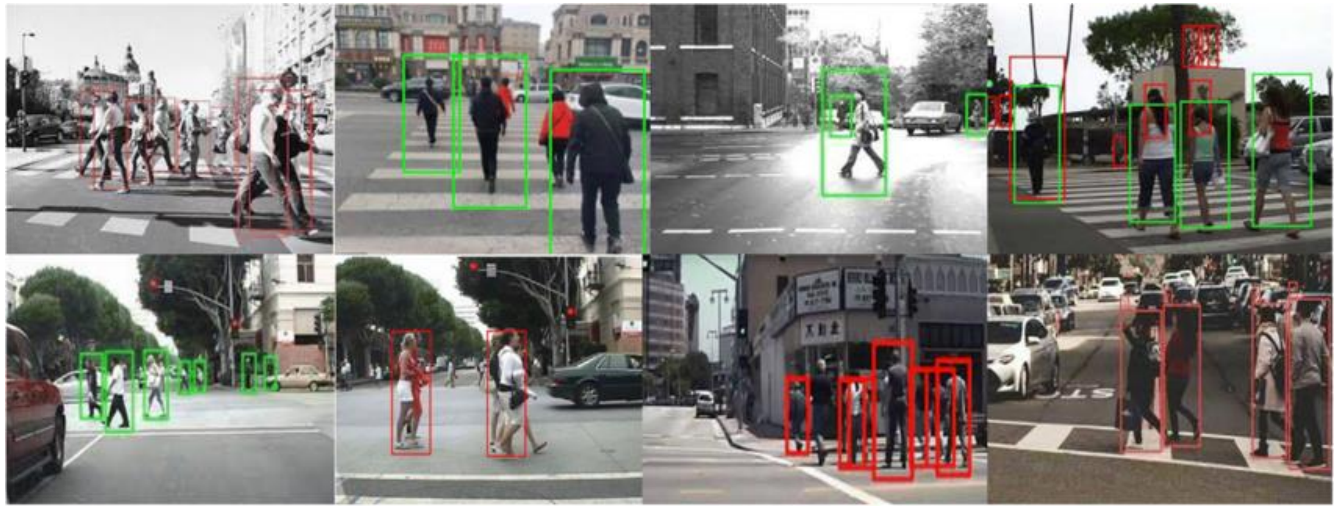
CNN - motivação

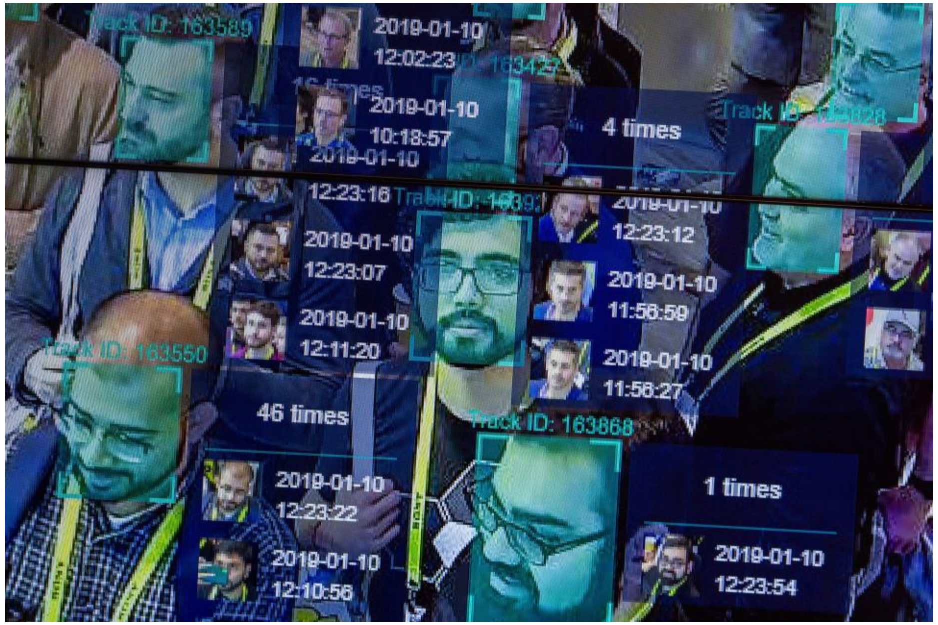
A vantagem das CNNs sobre outros algoritmos (SVM, K-NN, Random-Forest e outros) é que elas aprendem as melhores características para representar os objetos nas imagens, são muito versáteis e têm uma alta capacidade de generalização.

Por isso atualmente as CNNs estão presentes nas mais diversas áreas

CNN - motivação

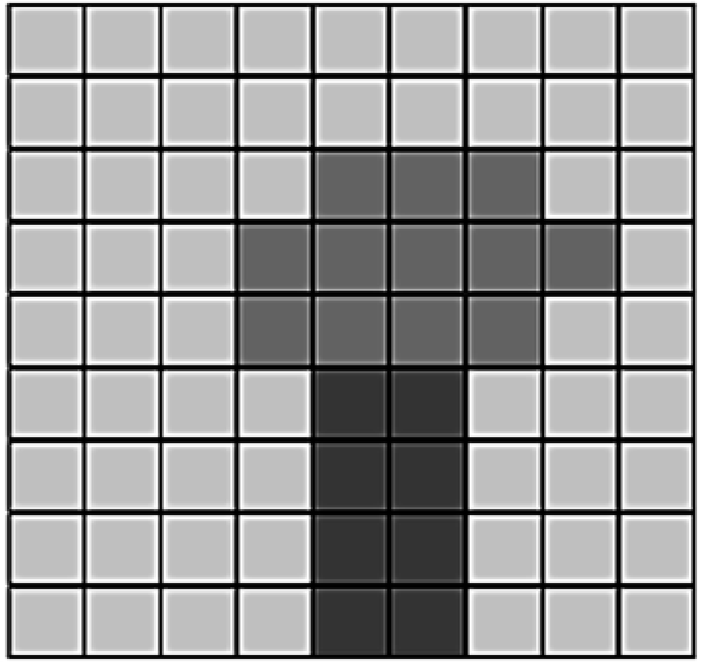




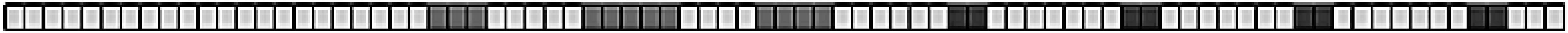


CNN – tudo começa com imagens

Processar este tipo de dado é complexo e custoso. Usando redes neurais clássicas, por exemplo, temos alguns problemas. O primeiro é o custo computacional pois muitas variáveis são geradas ao vetorizar as imagens:

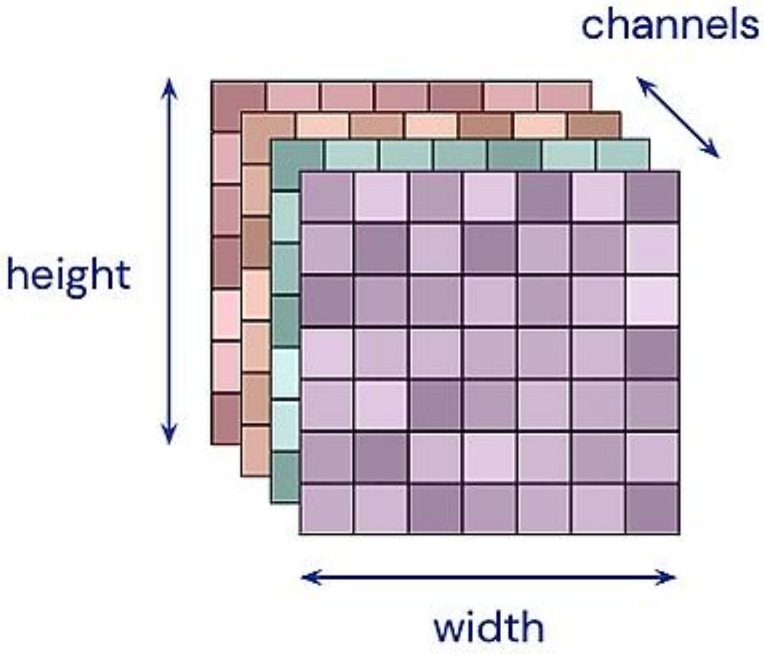


Contando da esquerda para a direita podemos fazer a cada linha:



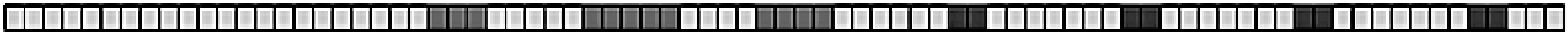
CNN – tudo começa com imagens

Uma imagem pode ser entendida como um Tensor de dimensões largura de pixel x altura de pixel x canais.



CNN – tudo começa com imagens

Contando da esquerda para a direita podemos fazer a cada linha:

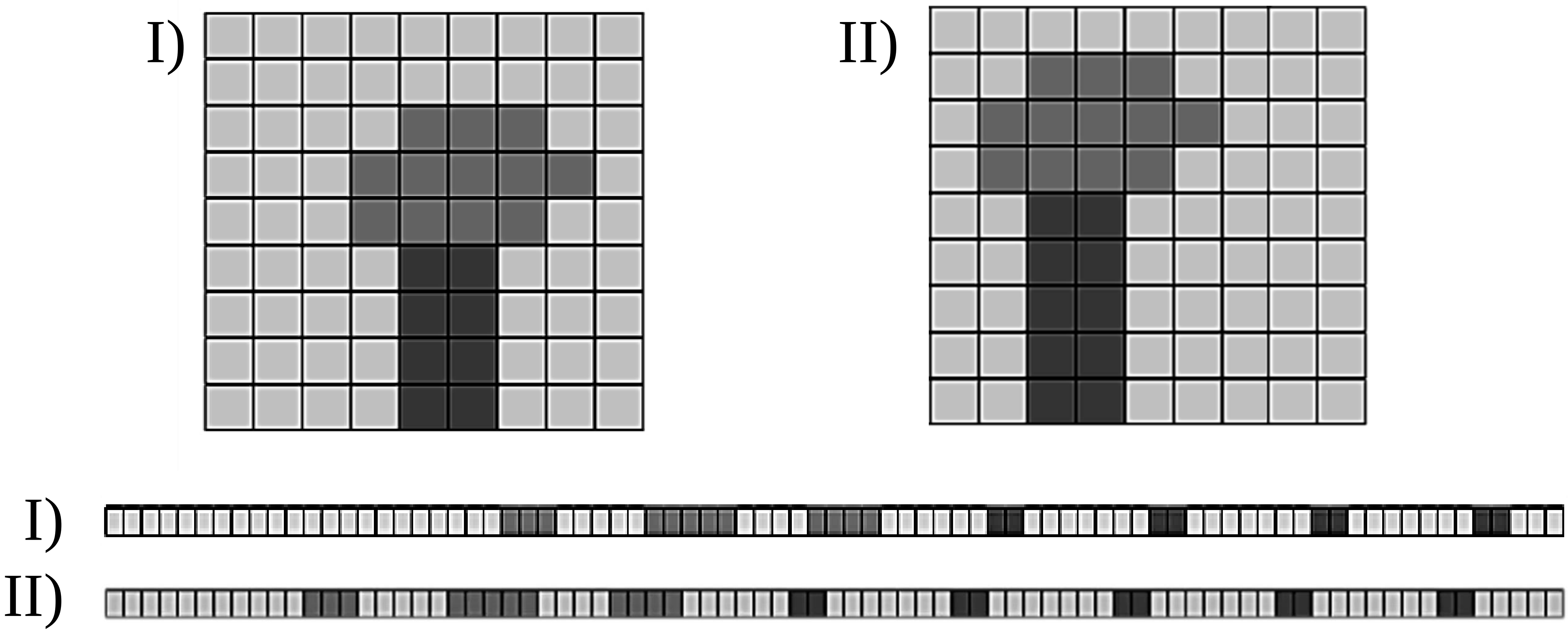


Uma imagem de qualidade HD (*high definition*) pode ter 1280x720 pixels. Isto decomposto nas escalas de cores tradicionais, equivale a um vetor de 3x921600 variáveis (RGB), 2x921600 na escala P&B e 921600 em tons de cinza. Cada variável tem um peso associado (isso só na camada 1).

Seria necessária uma rede com milhares de hiperparâmetros, cujo crescimento em complexidade é exponencial a medida que se aumentam as camadas para tornar o treinamento viável ("Curse of high dimensinality")

CNN – tudo começa com imagens

Outro problema grave ao utilizar redes clássicas para processar imagens vetorizadas é a independência transacional



Convolução

CNN – Processo de convolução

Uma abordagem empregando uma arquitetura diferente é necessária. Dai surgem as redes neurais convolucionais, que integram filtros *kernel* ao processo de convolução para lidar de forma adequada com as imagens

Convolução pode ser grosseiramente entendida como uma operação matemática entre duas funções para produzir uma terceira, que expressa o quanto o formato da primeira é modificado pela segunda

Já de forma mais precisa, é um operador linear que, a partir de duas funções iniciais definidas em um mesmo domínio, resulta numa terceira que mede a soma do produto dessas funções ao longo de uma região subentendida pela superposição delas em função de um deslocamento pré-fixado

CNN – Processo de convolução

A convolução 1D entre duas funções f e g definidas no domínio contínuo é:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(u)g(x-u)du$$

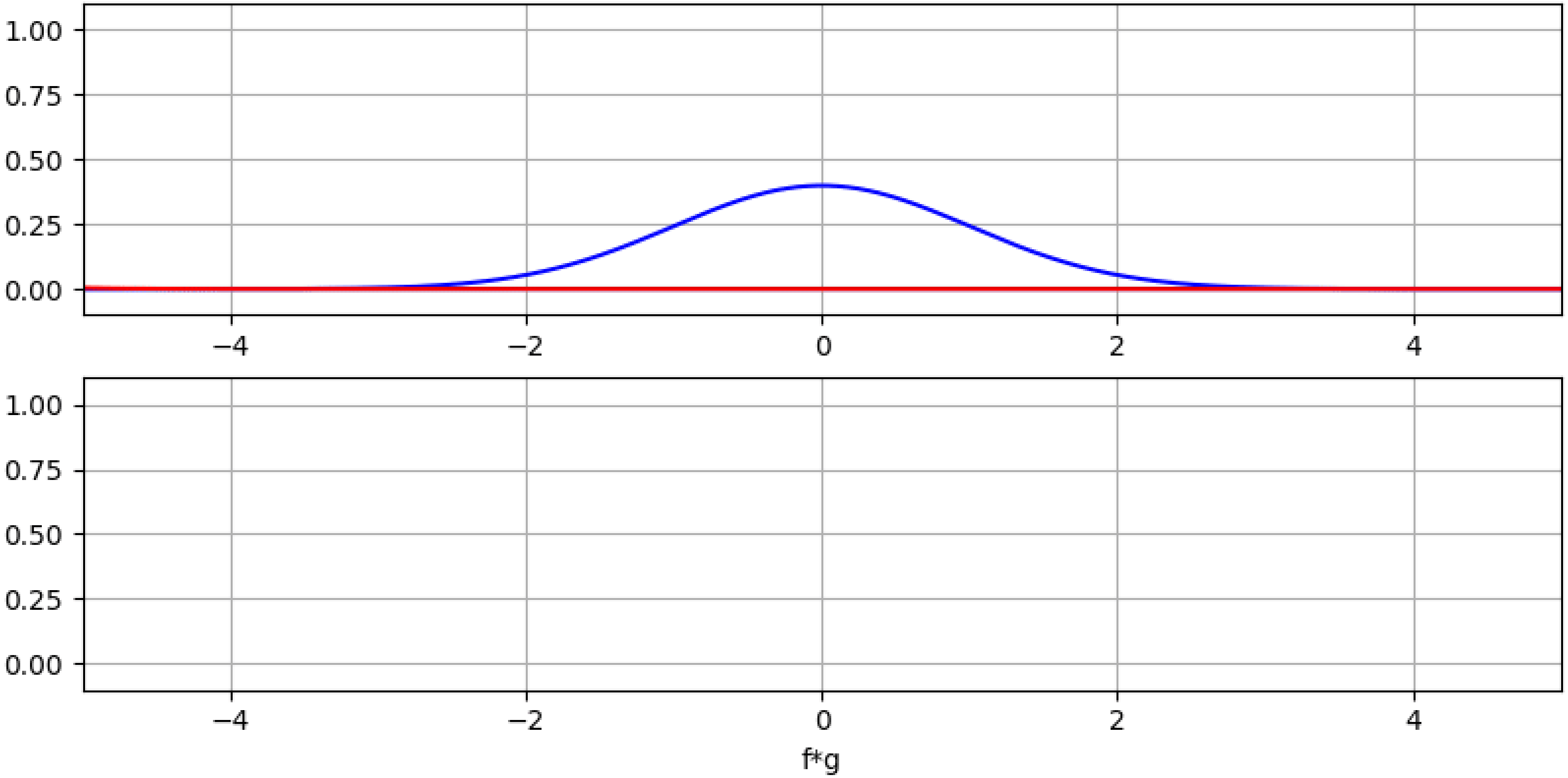
O u é a variável de integração (ou variável "fantasma"), que percorre todos os valores do domínio ($-\infty$ a ∞) calculando a área sob o produto de $f * g$, que varre cada valor de x.

Imagine que $f(.)$ seja uma função avaliada e que para cada valor de $f(x)$, u varia dentro dos limites de integração para permitir que $g(x-u)$ "deslize" sobre $f(.)$. A integral nos fornece a soma total da contribuição de g em cada x.

CNN – Processo de convolução

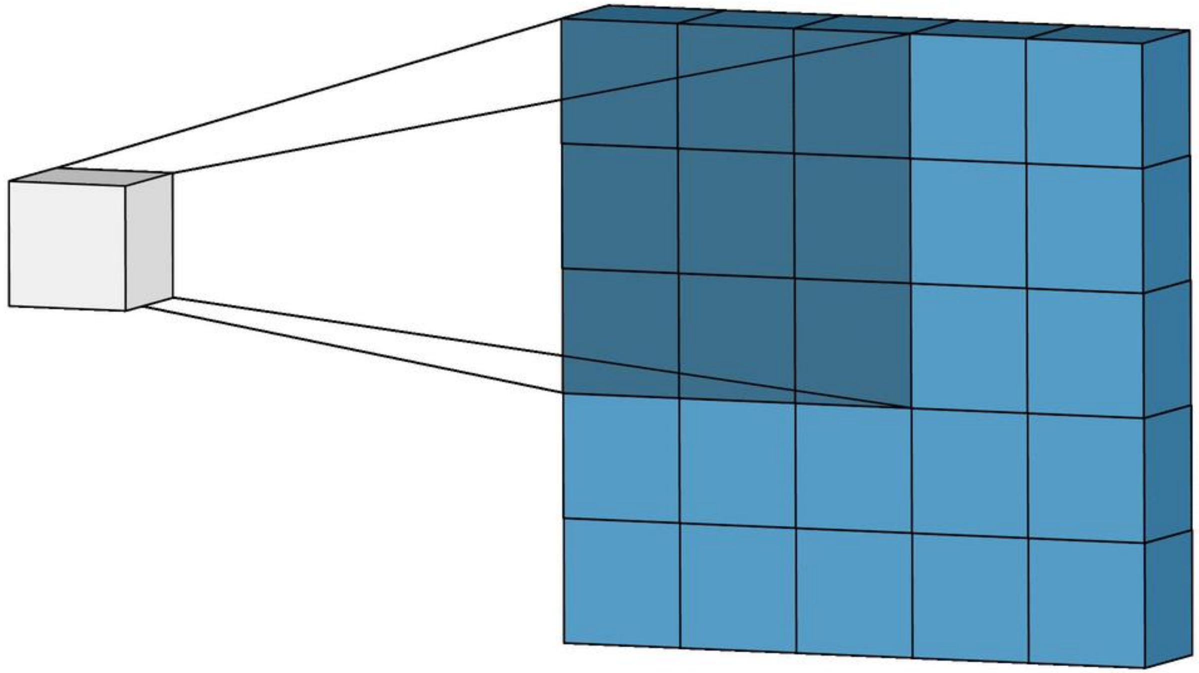
Analogamente, a convolução 1D entre duas funções f e g definidas no domínio discreto pode ser definida como:

$$(f * g)(x) = \sum_{u=-\infty}^{\infty} f(u)g(x-u)$$



CNN – Camada de Convolução

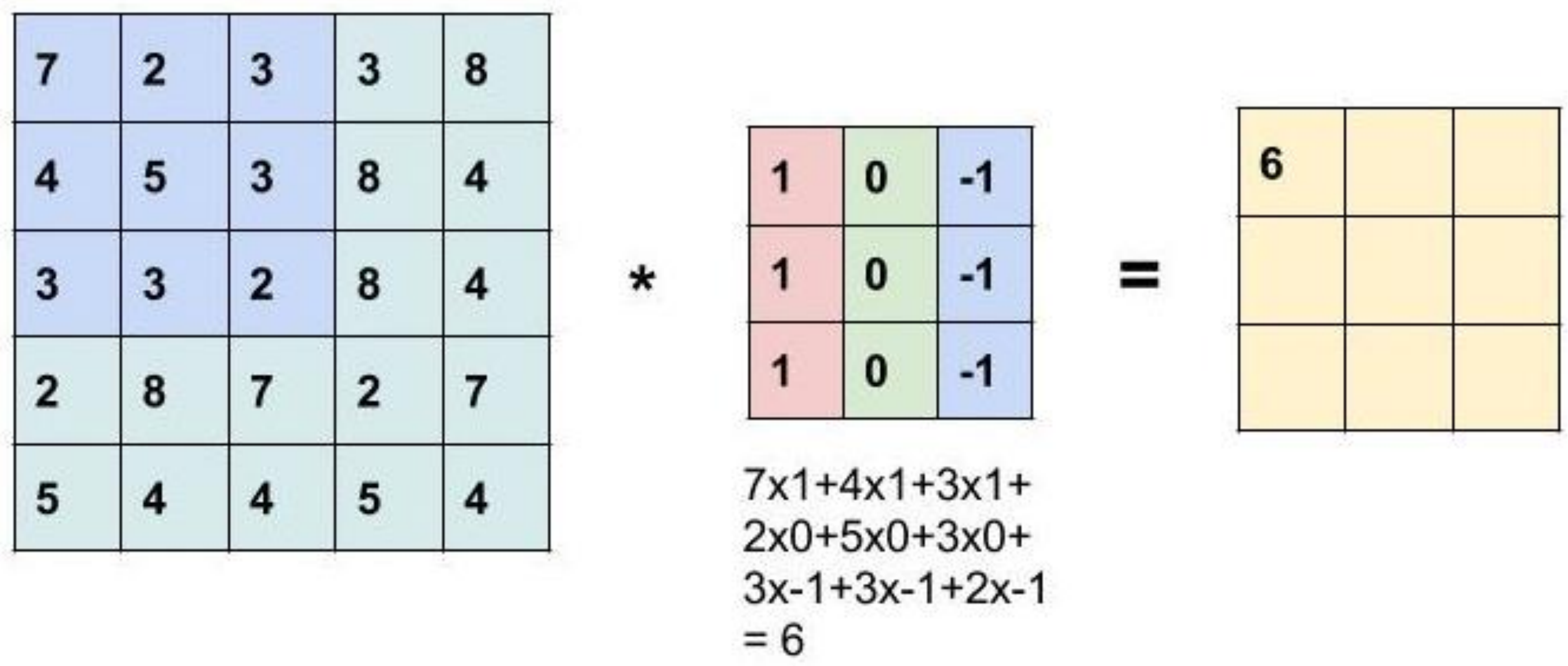
As CNNs implementam camadas de convolução para extrair features de alto nível dos dados. Vamos começar pela ideia em 2D:



Empregando um dados 2D (imagem em tons de cinza por ex) como a matriz de input (ou tensor de ordem 2), um filtro (kernel) é utilizado para o processo de convolução, deslizando pela imagem e produzindo um valor de saída para cada passo (stride).

CNN – Camada de Convolução

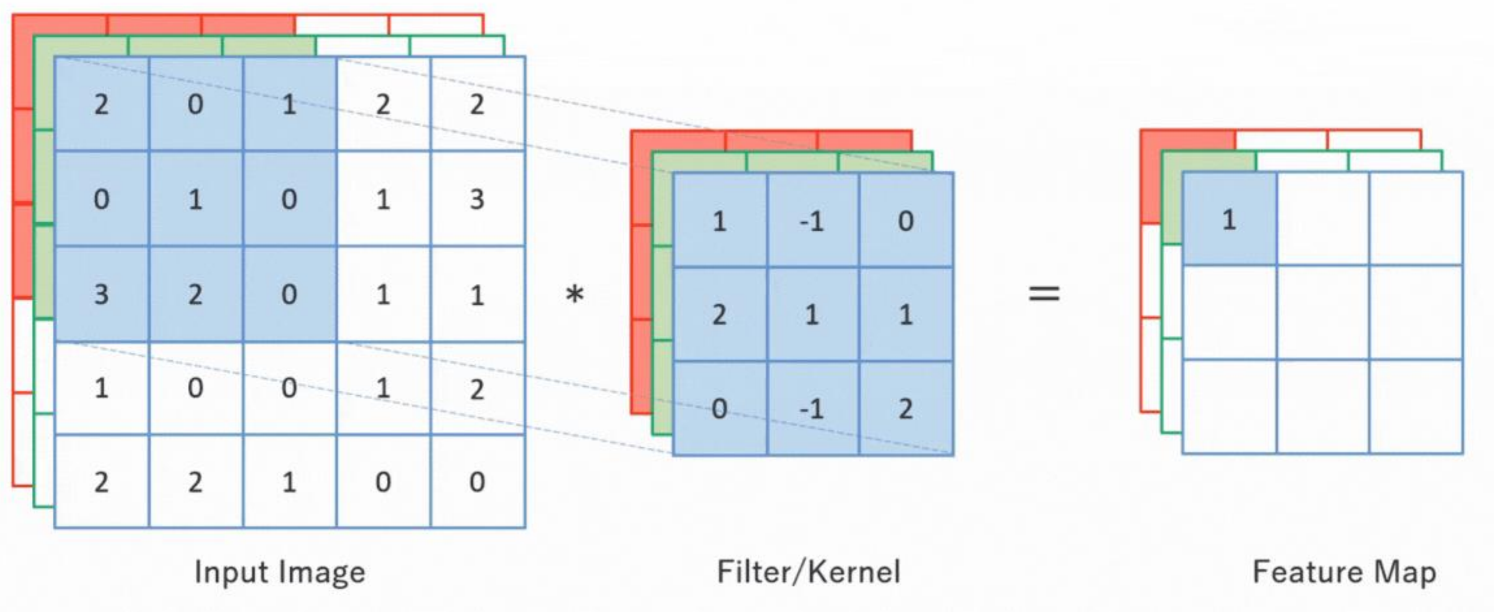
O tamanho do kernel e o stride são pré-ajustados. A área que o kernel cobre (o grid do kernel) é chamado de campo receptivo. Ele carrega consigo uma série de pesos (hiperparametros) que são otimizados durante o treinamento



CNN – Camada de Convolução

Além disso, também podemos controlar o numero de filtros (canais) do kernel, que deve ter, no mínimo, o mesmo numero de canais (profundidade) dos dados de input

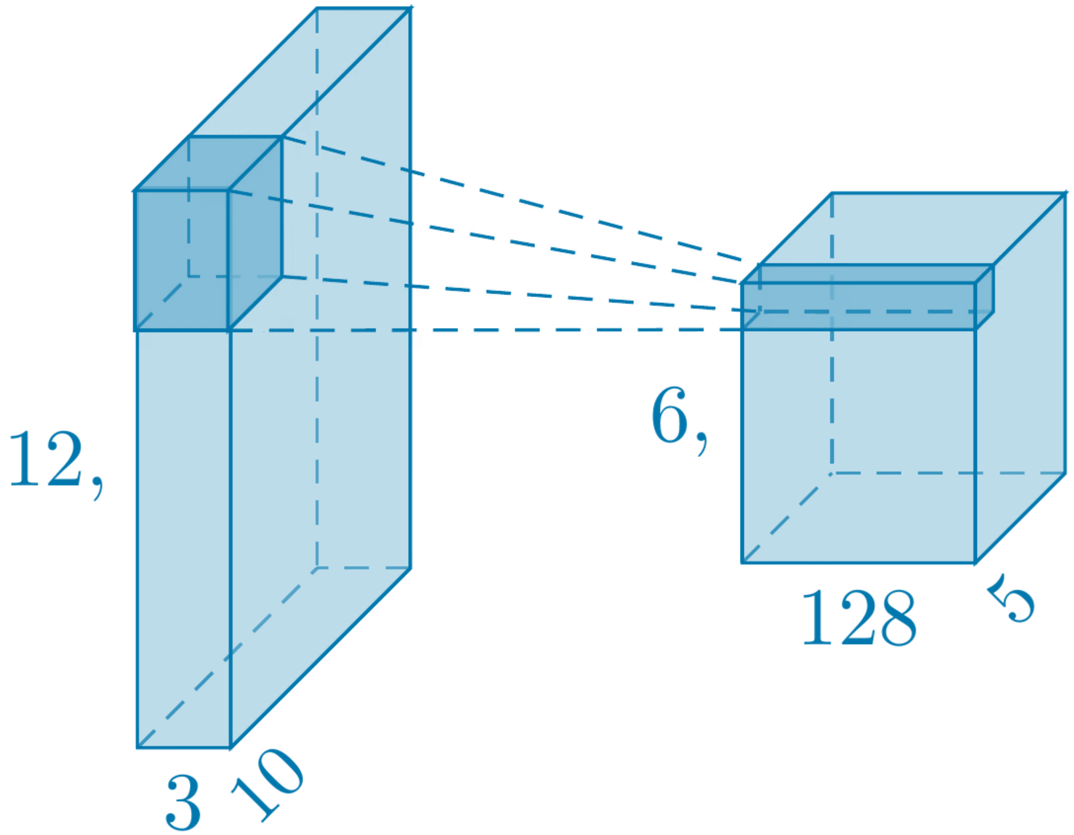
No caso de matrizes 2D (altura x largura) a profundidade é 1. Mas outros dados tensoriais (*e.g.* imagens RGB) podem ter múltiplas dimensões na profundidade



Assim são gerados os *feature/activation maps* (um para cada canal/profundidade do kernel). Eles são as saídas passadas adiante para as camadas subsequentes

CNN – Camada de Convolução

O kernel pode aumentar, diminuir ou manter as dimensões do tensor que ele convoluiu. Depende da dimensão adotada (largura x altura x profundidade):

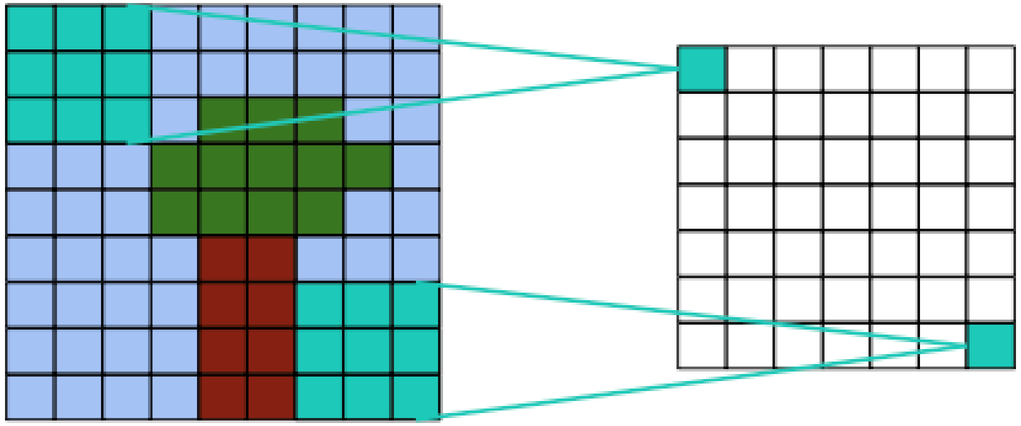


Ex: Um kernel de campo receptivo de dimensões 3 x 3 x 1 (largura x altura x canais) com passo de 1 aplicado a uma matriz 6 x 6 x 1 geraria um *feature map* de tamanho 4 x 4 x 1

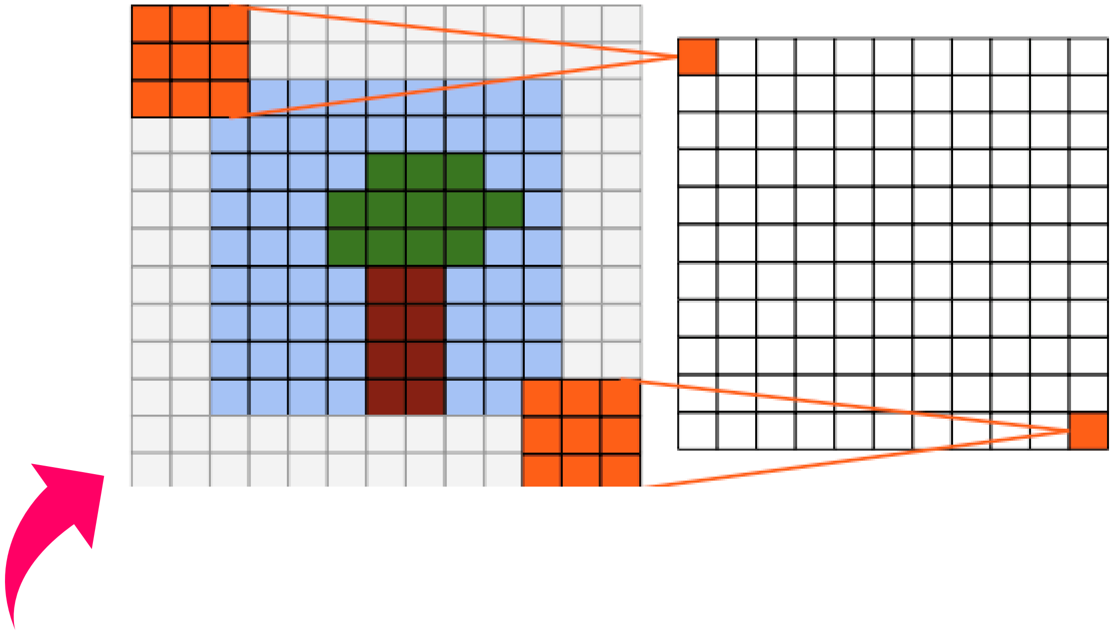
CNN – Camada de Convolução

Nesta perspectiva, diferentes estratégias de convolução podem ser adotadas:

Diminuição
(output = input – kernel +1)



Aumento
(output = input + kernel -1)

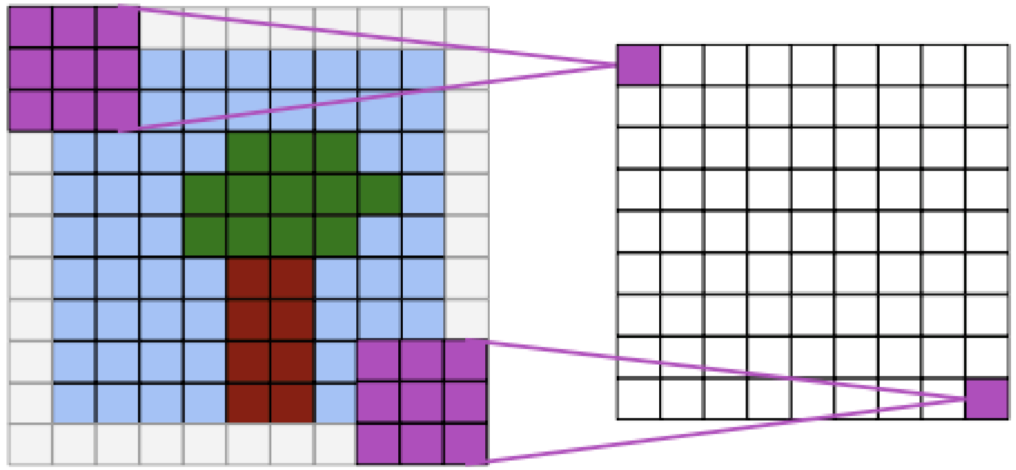


Nesse sentido, a técnica de *padding* (preenchimento) é utilizada. O mais conhecido é o *zero padding*, mas existem também *average padding*, *same padding*, entre outros...

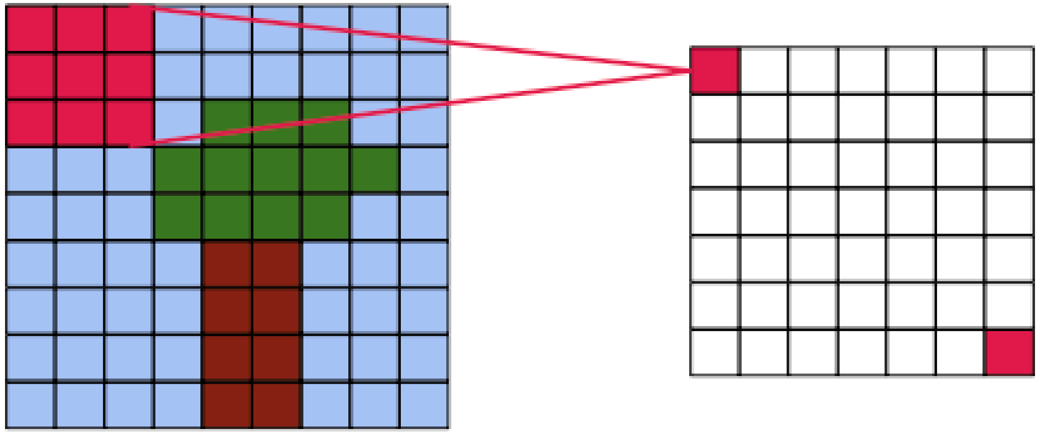
CNN – Camada de Convolução

Nesta perspectiva, diferentes estratégias de convolução podem ser adotadas:

Equivalente
(output = input)

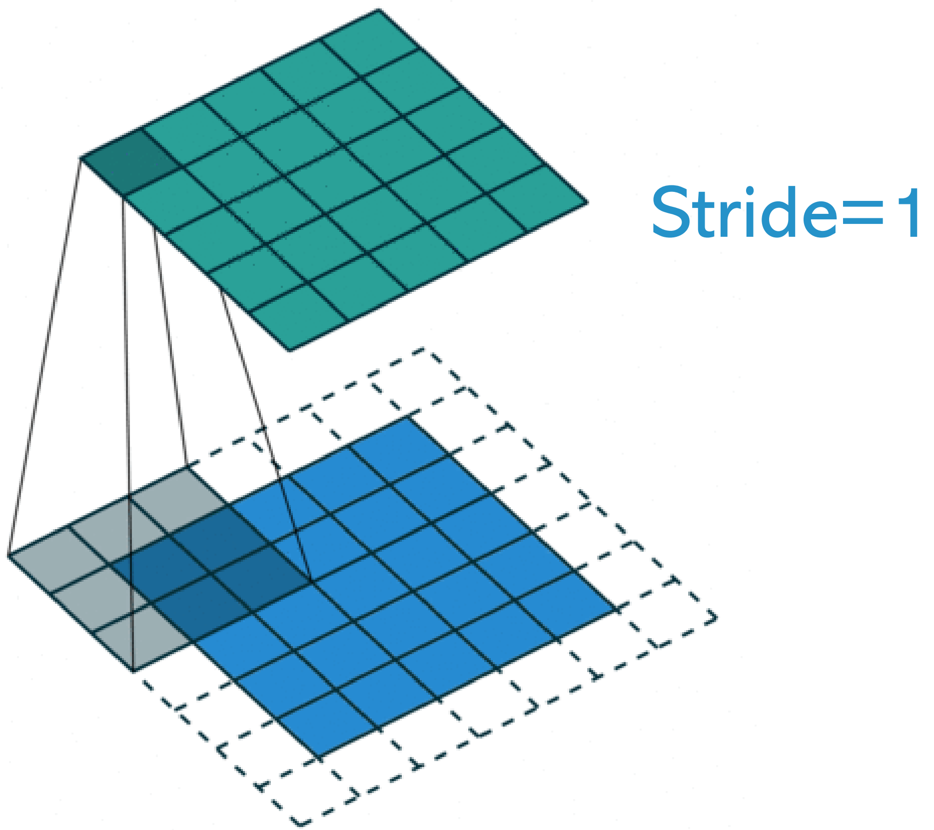


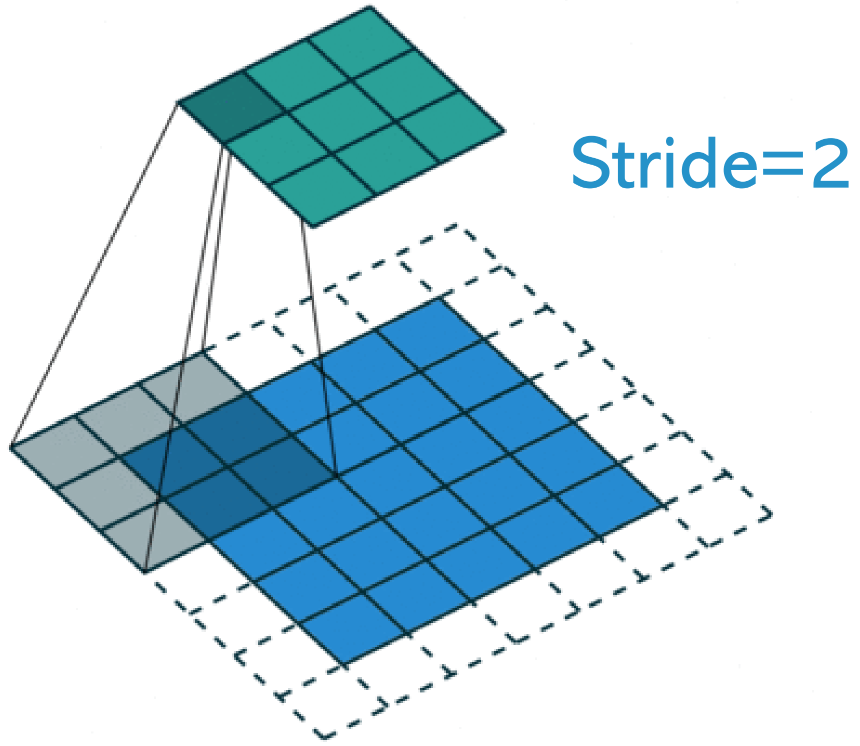
Multiplas conexões



CNN – Camada de Convolução

O passo do kernel também tem influência na dimensão do *feature map* gerado



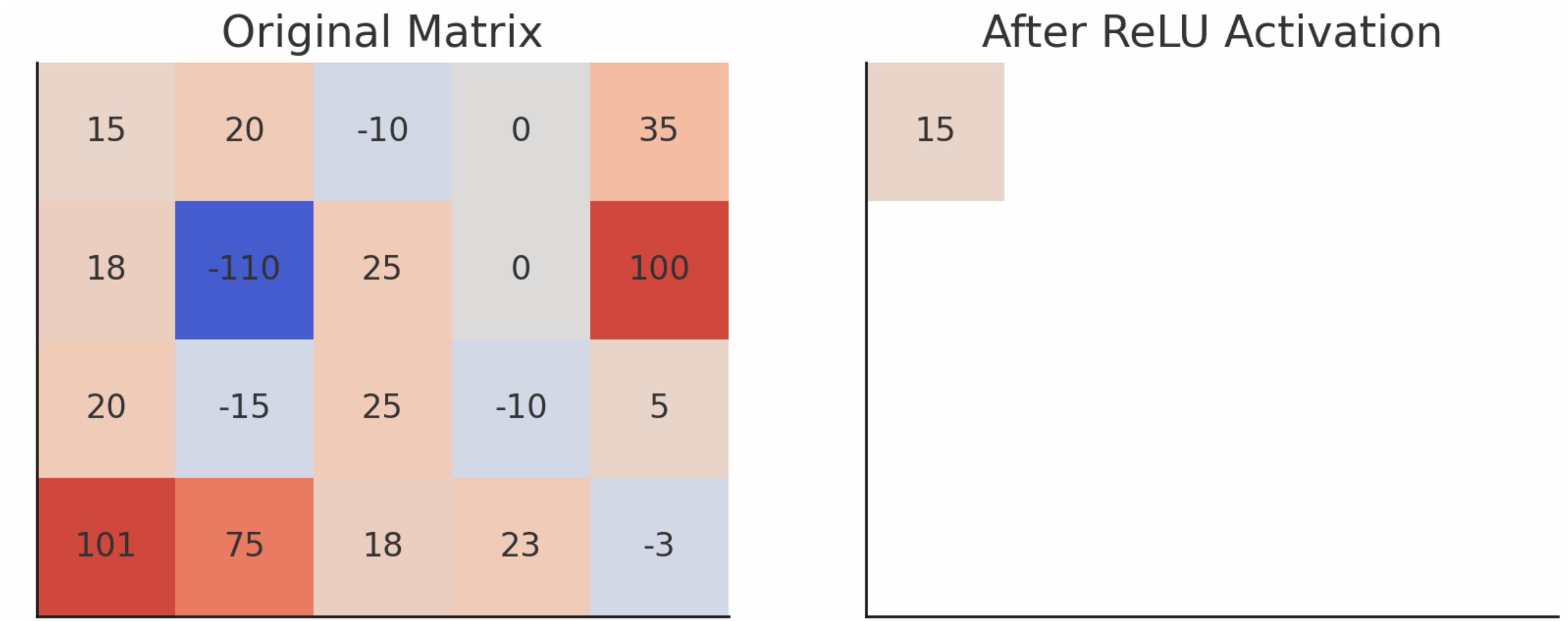


Portanto, a combinação do stride e das dimensões do kernel permite controlar o tamanho do *feature map* gerado, e portanto a dimensão de cada camada de convolução da rede

CNN – Camada de Convolução

Após gerados, os *feature maps* passam por funções de ativação para o acréscimo de não-linearidade no processamento (importante para convergência da rede e captura de padrões não-lineares nos dados)

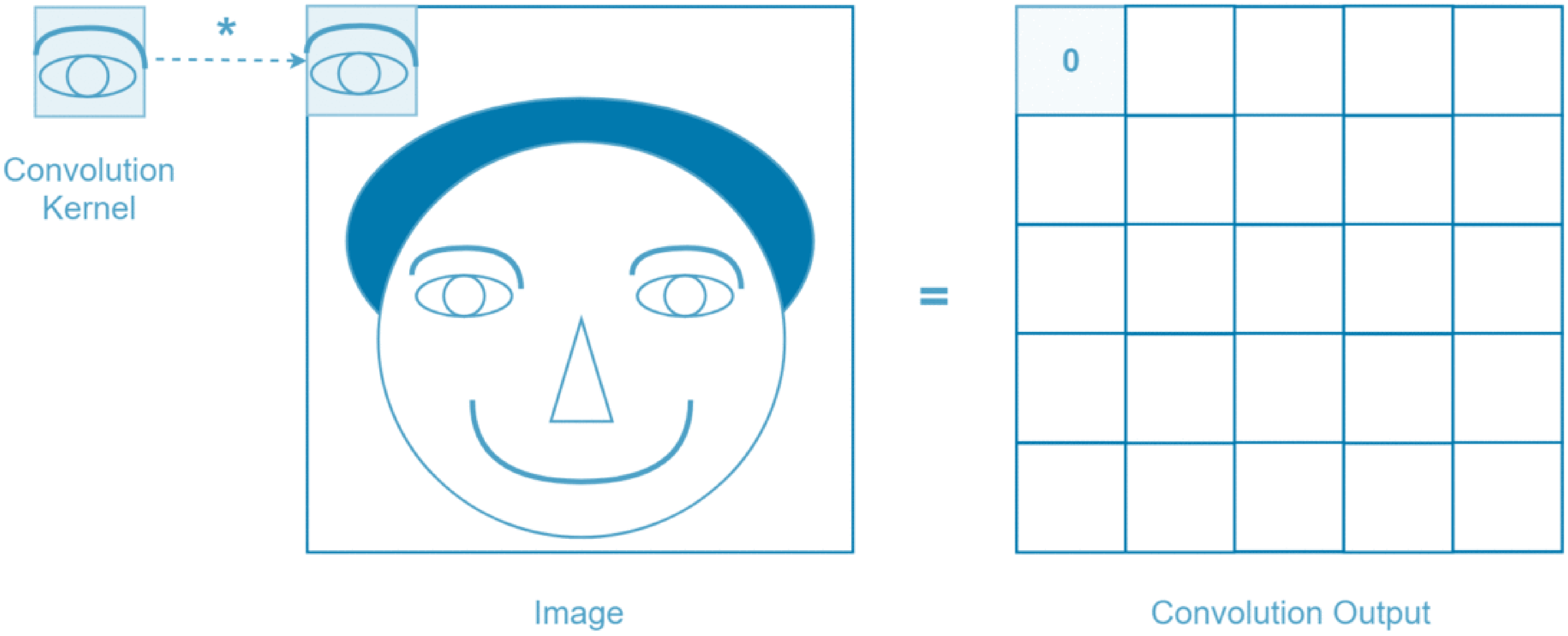
Entre as tradicionais (*tanh*, *sigmoid*, etc..), a ReLU é a mais utilizada



zera valores negativos

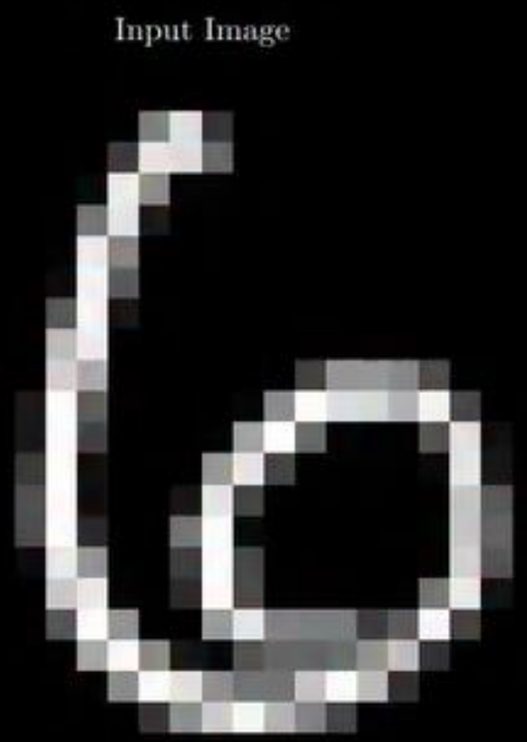
CNN – Camada de Convolução

Os pesos de cada kernel de cada são compartilhados por todos os dados de input



Logo, eles são ajustados para a aprender a identificar os padrões locais relacionados ao target em qualquer parte dos dados, apresentando independência translacional

Animation of a Convolution

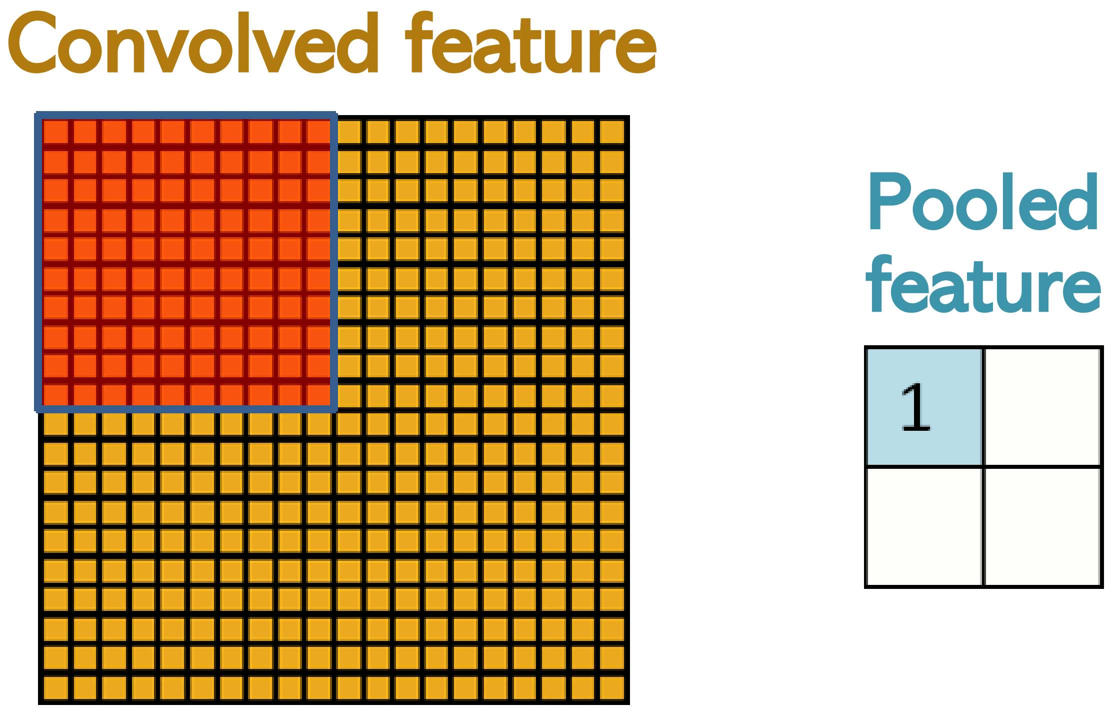


Camada de *Pooling*

CNN – Camada de *Pooling*

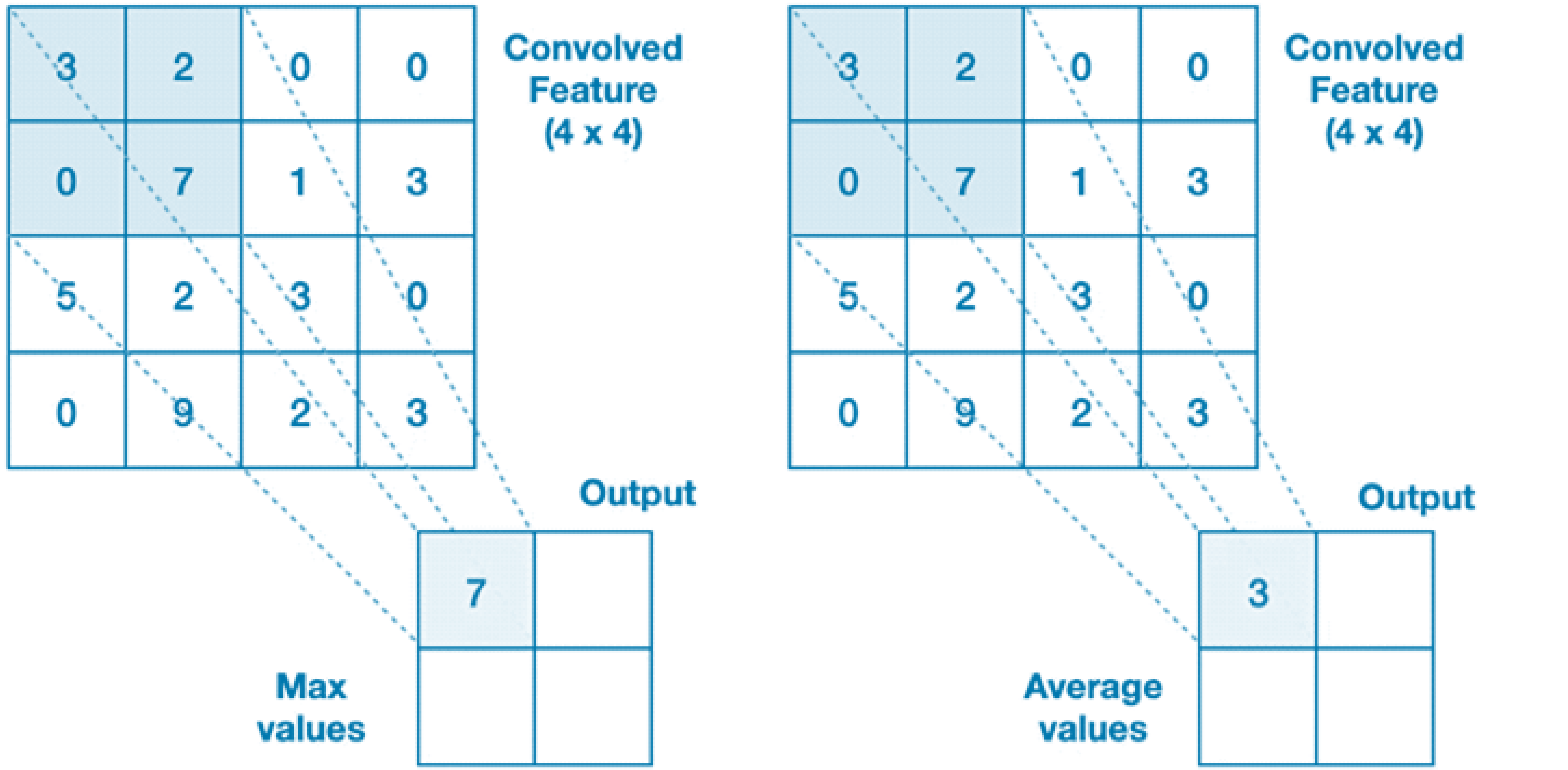
Embora o problema da independência translacional tenha sido resolvido, ainda precisamos lidar com os números elevados de hiperparametros treináveis

a ideia da camada de ***Pooling*** é reduzir drasticamente a dimensão dos *feature maps* após cada camada de convolução. Esse processo, também chamado de ***downsampling***, permite acelerar os cálculos computacionais da rede



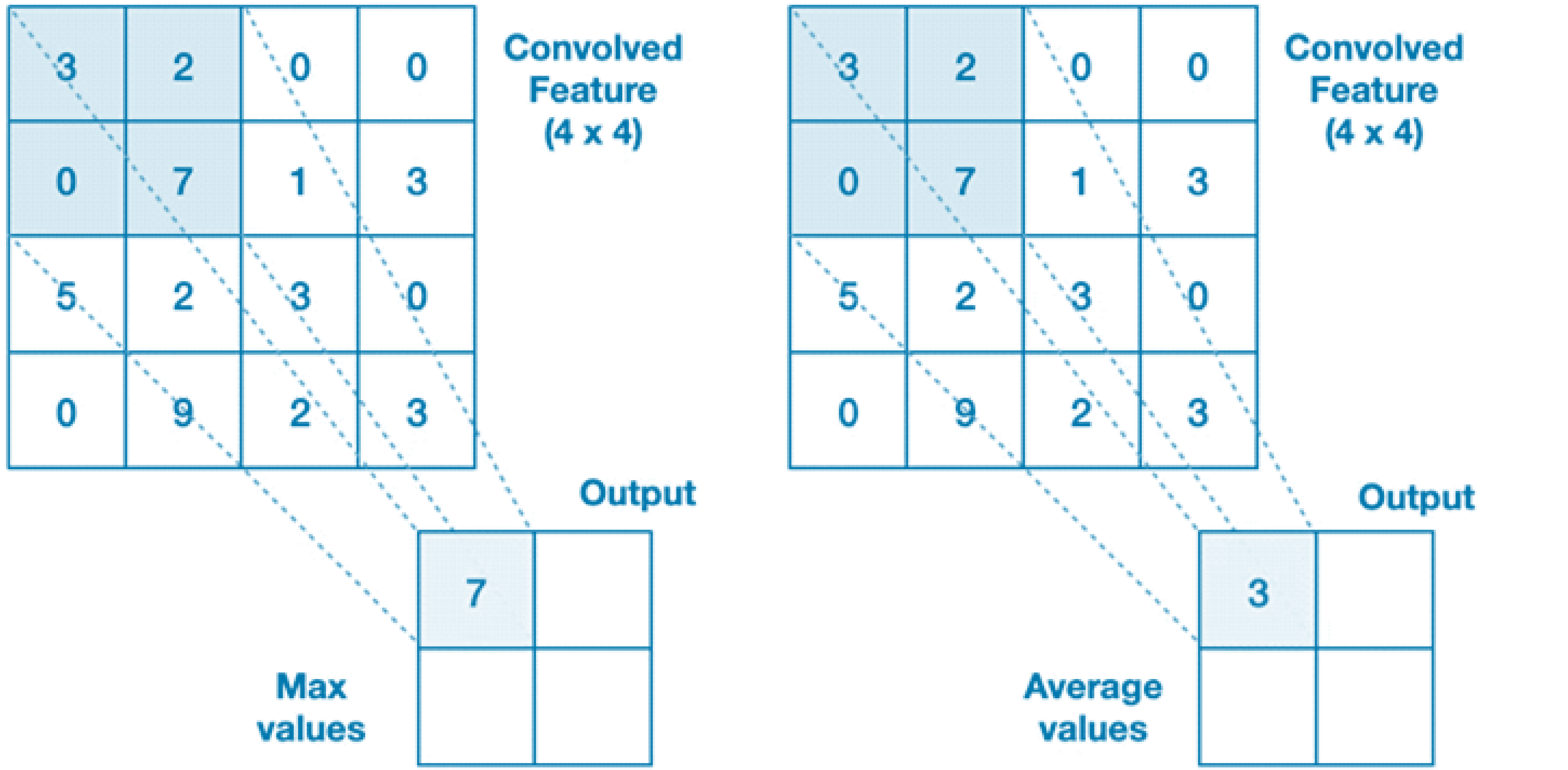
CNN – Camada de *Pooling*

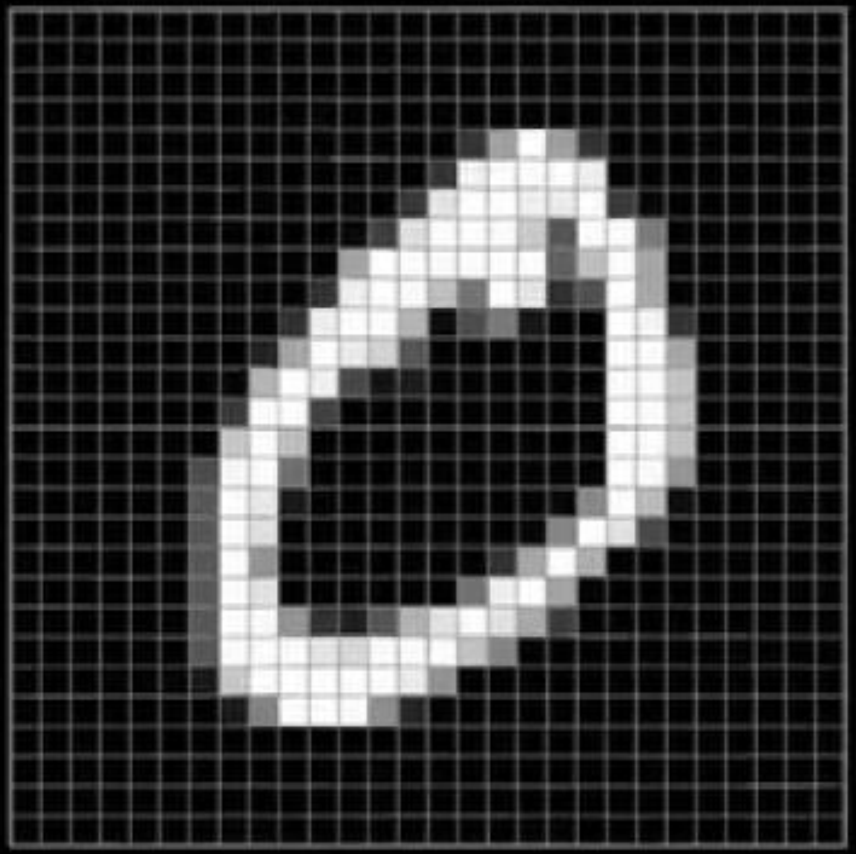
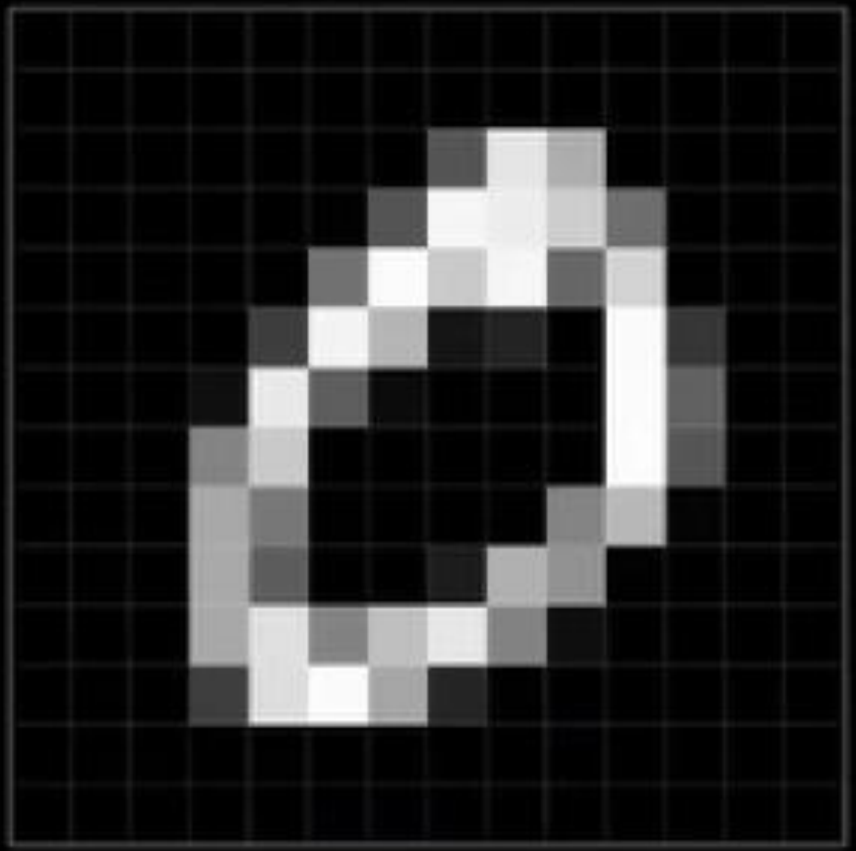
Os mais utilizados são o *MaxPooling* e o *AveragePooling*, que reduzem a dimensionalidade selecionando os maiores e médios valores a cada passo deslizando sobre features maps, respectivamente



CNN – Camada de *Pooling*

Analogamente à convolução, define-se um tamanho de área (largura x altura) e um stride, mas desta vez, ao invés de extrair informações (como os kernels), o objetivo é comprimir as informações já extraídas

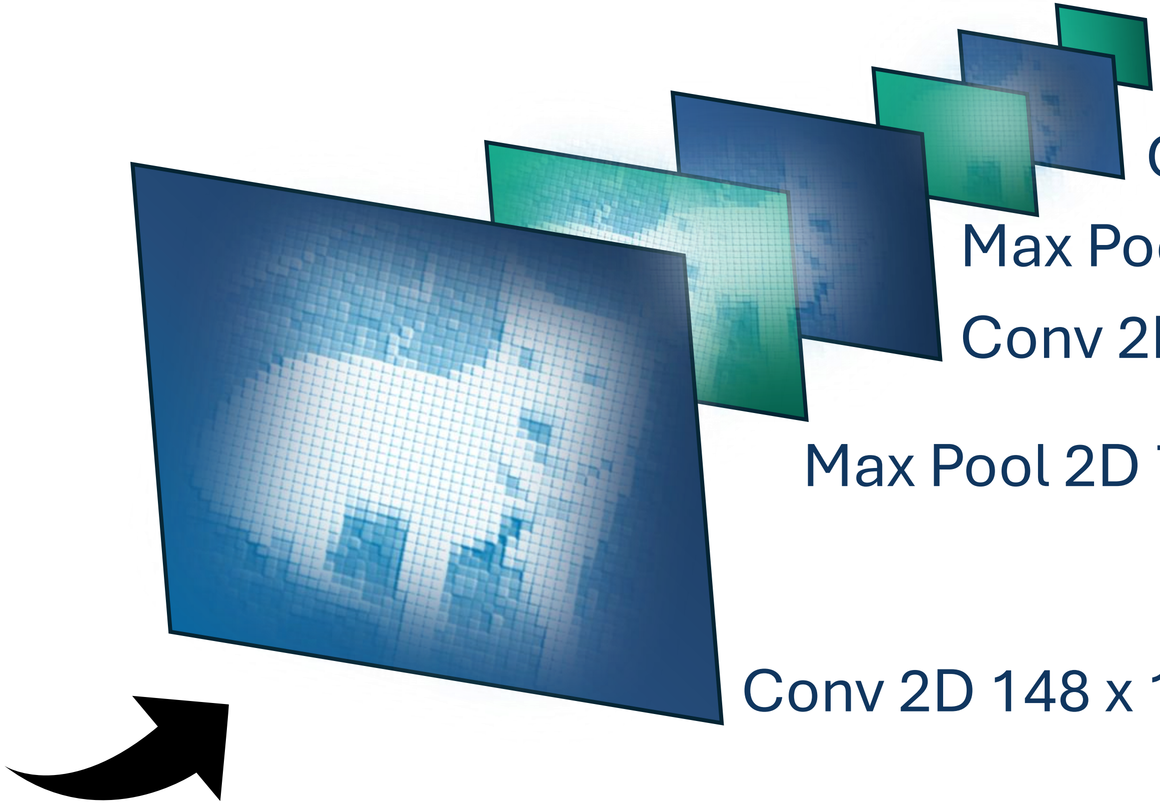


Visualizando *os feature maps*

Os ***features maps*** extraem as principais características, úteis para o modelo. Eles são possíveis de serem visualizados, através do plot do resultado da sua aplicação nos dados (após o treinamento). Ex: CNN de classificação treinada com imagens de animais:

Original 150 x150 x 3



Max Pool 2D 17 x 17 x 32

Conv 2D 34 x 34 x 32

Max Pool 2D 36 x 36 x 16

Conv 2D 72 x 72 x 16

Max Pool 2D 74 x 74 x 8

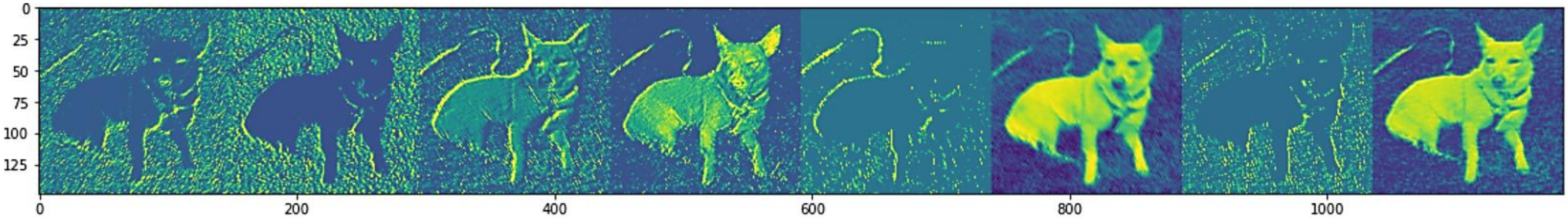
Conv 2D 148 x 148 x 8

Pooling size 2 x 2 (stride=1)

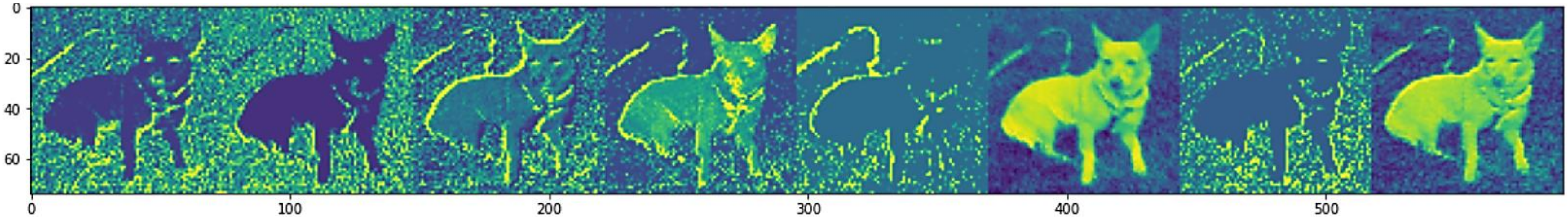
Kernel size 3 x 3 (stride=1)

link

Conv 2D 148 x 148 x 8

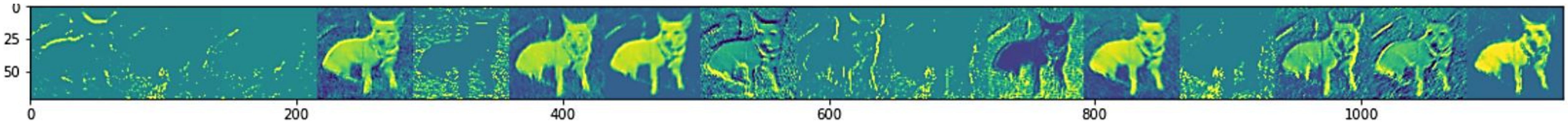


Max Pool 2D 74 x 74 x 8

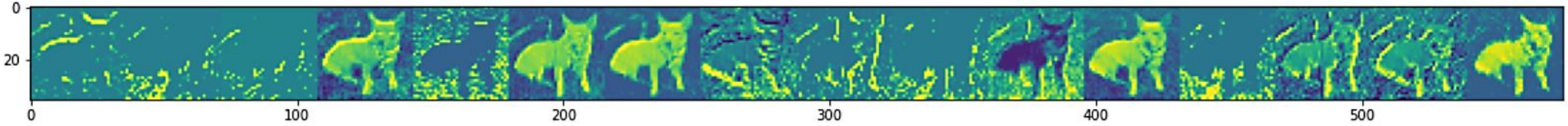


link

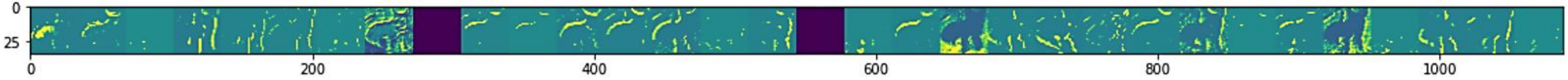
Conv 2D 72 x 72 x 16



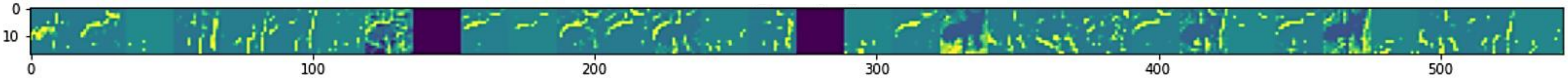
Max Pool 2D 36 x 36 x 16



Conv 2D 34 x 34 x 32



Max Pool 2D 17 x 17 x 32

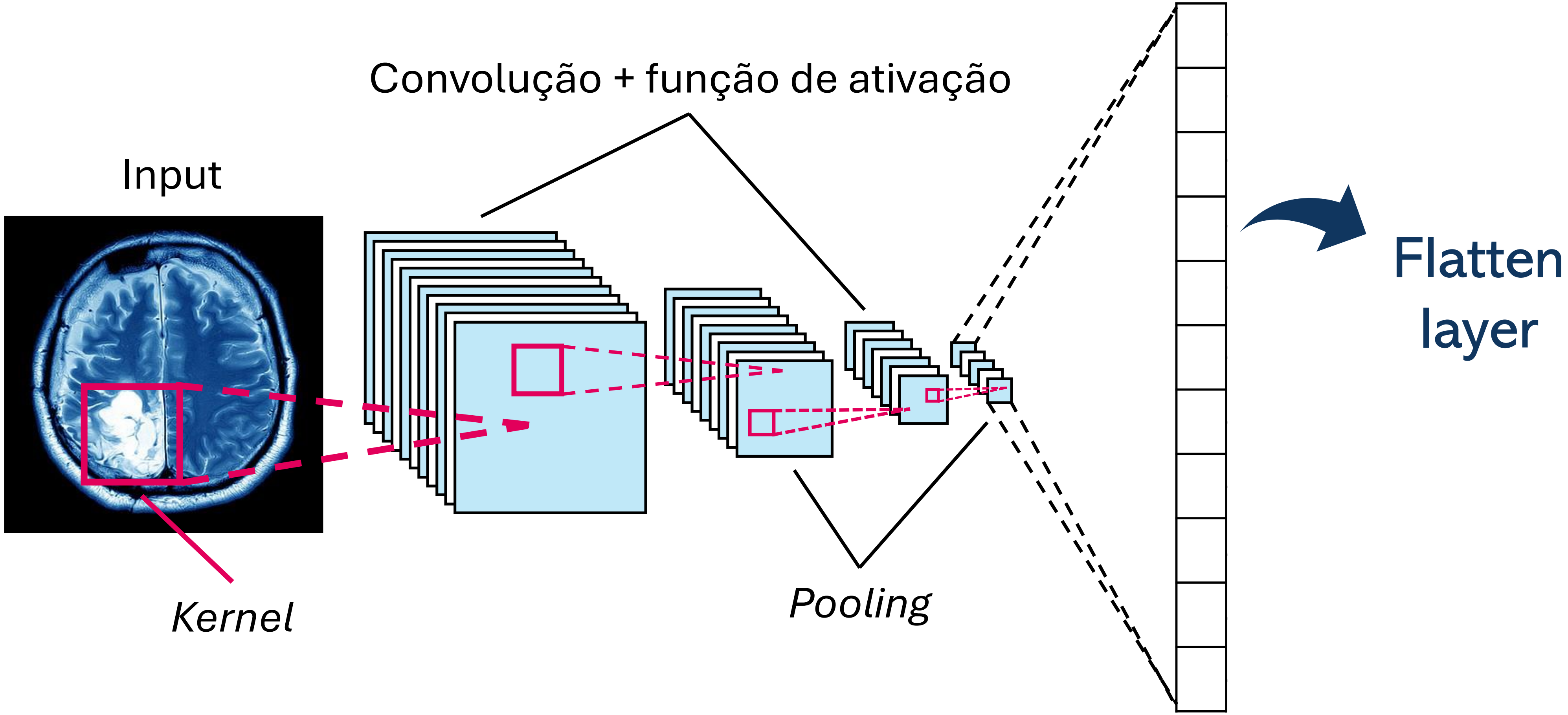


link

Camada de achatamento

CNN – Camada de achatamento

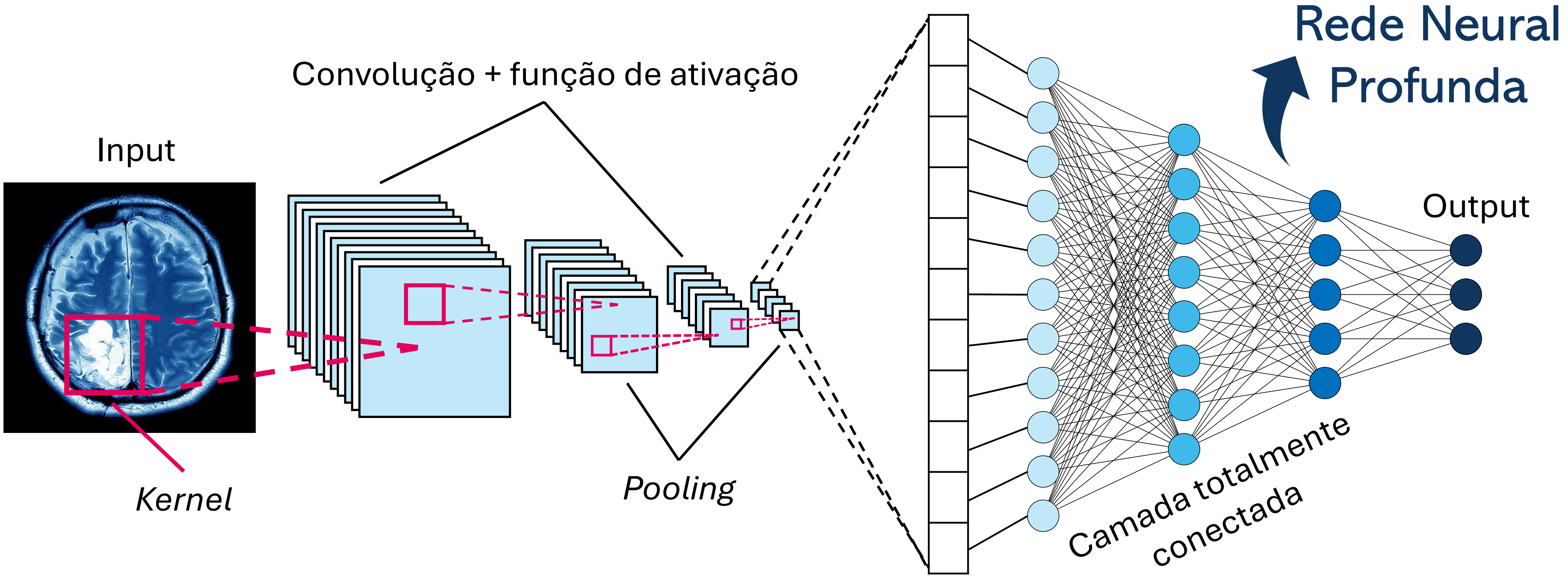
Após os dados passarem por diversas camadas de convolução e pooling eles são achatados em um vetor unidimensional (flatten layer) antes de seguirem para as próximas camadas da rede



Camada totalmente conectada

CNN – Camada totalmente conectada

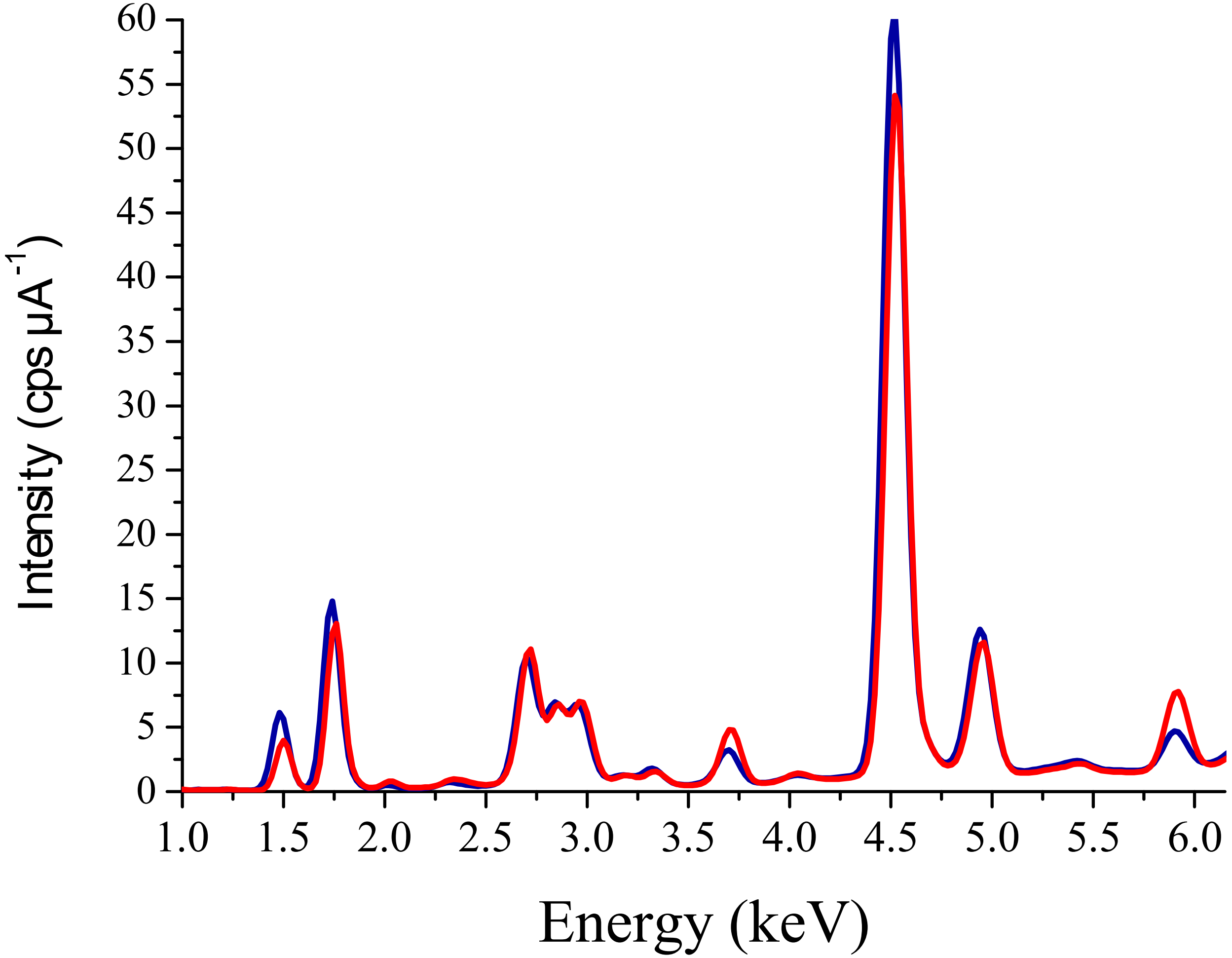
Após os dados passarem pelas camadas de convolução e *pooling* (extrair os *feature maps*) e serem achatados eles são processados através de **redes neurais profundas**, que são arquitetadas de acordo com o que se deseja prever/classificar



CNN 1D – lidando com espectros

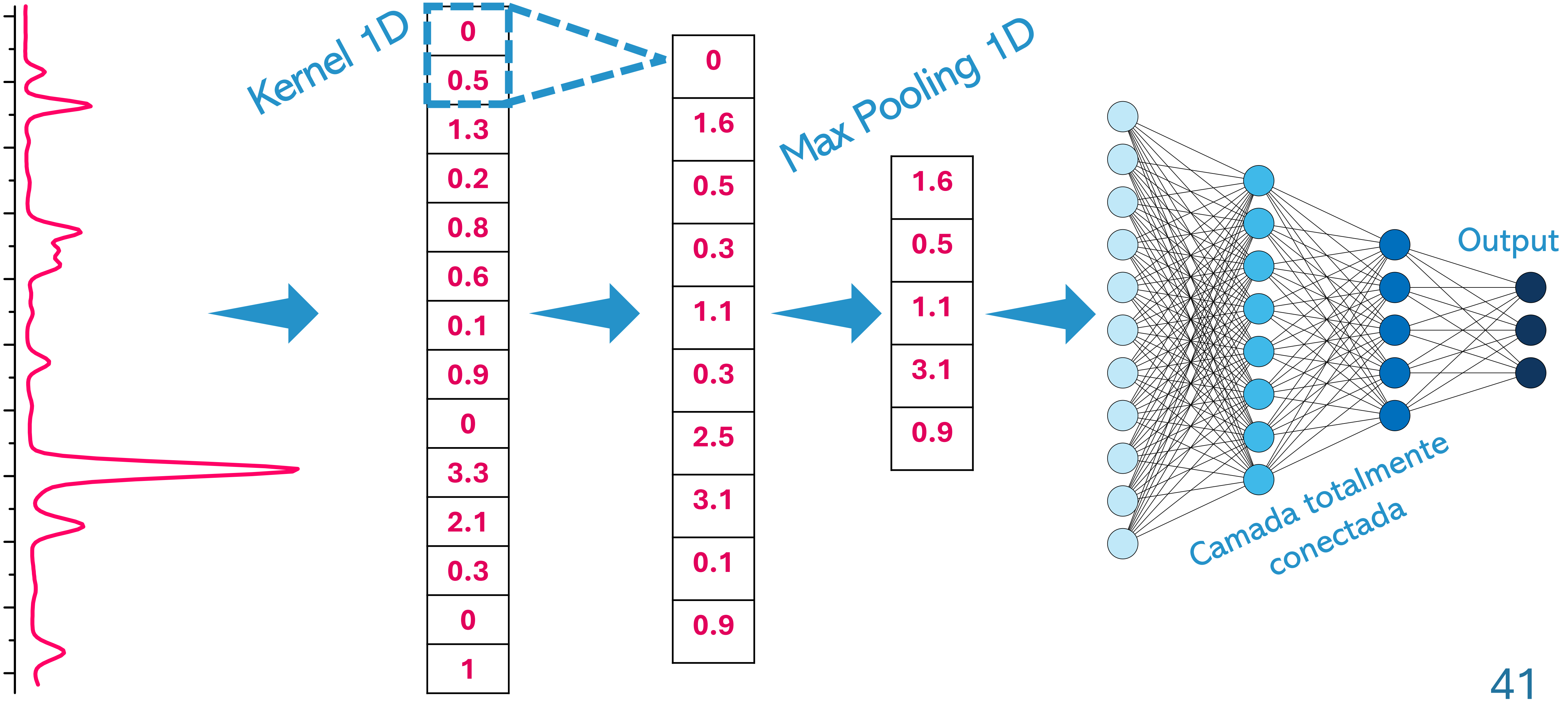
CNN 1D – lidando com espectros

O espectros de XRF, vis-NIR, GRS etc. tem como característica subjacente a unidimensionalidade. Ou seja, para cada amostra nós registramos uma única informação (*e.g.* contagem) por variável (*e.g.* energia ou comprimento de onda)



CNN 1D – lidando com espectros

Ex: processo de convolução e *pooling* unidimensional em uma CNN

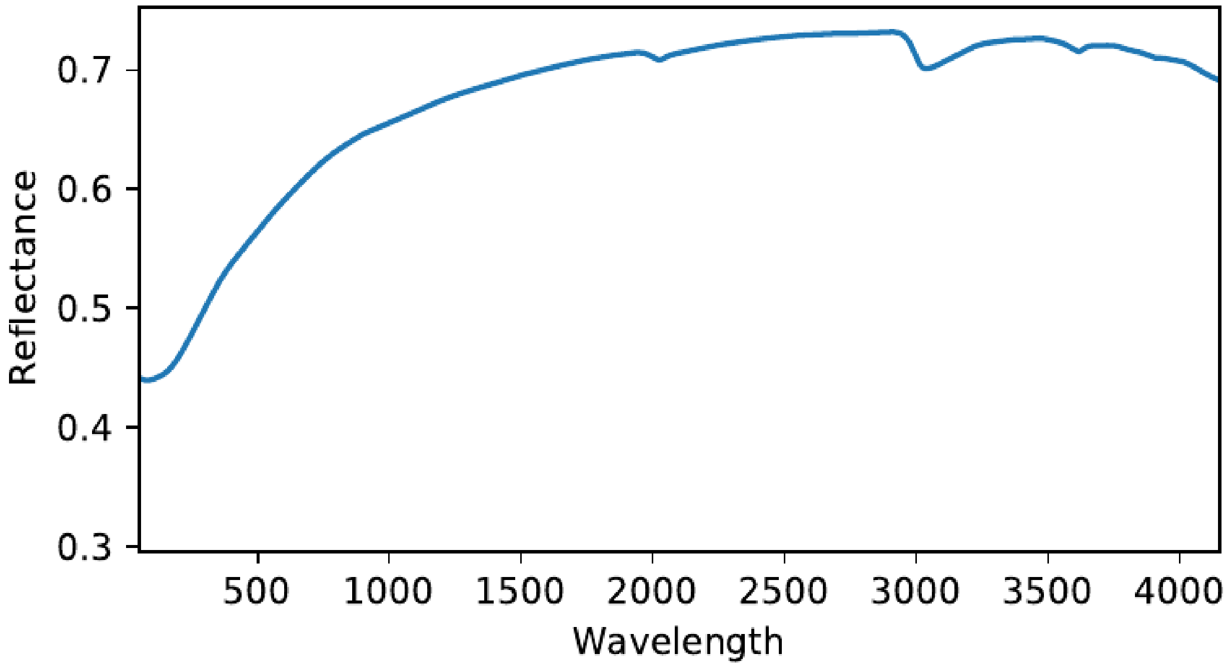


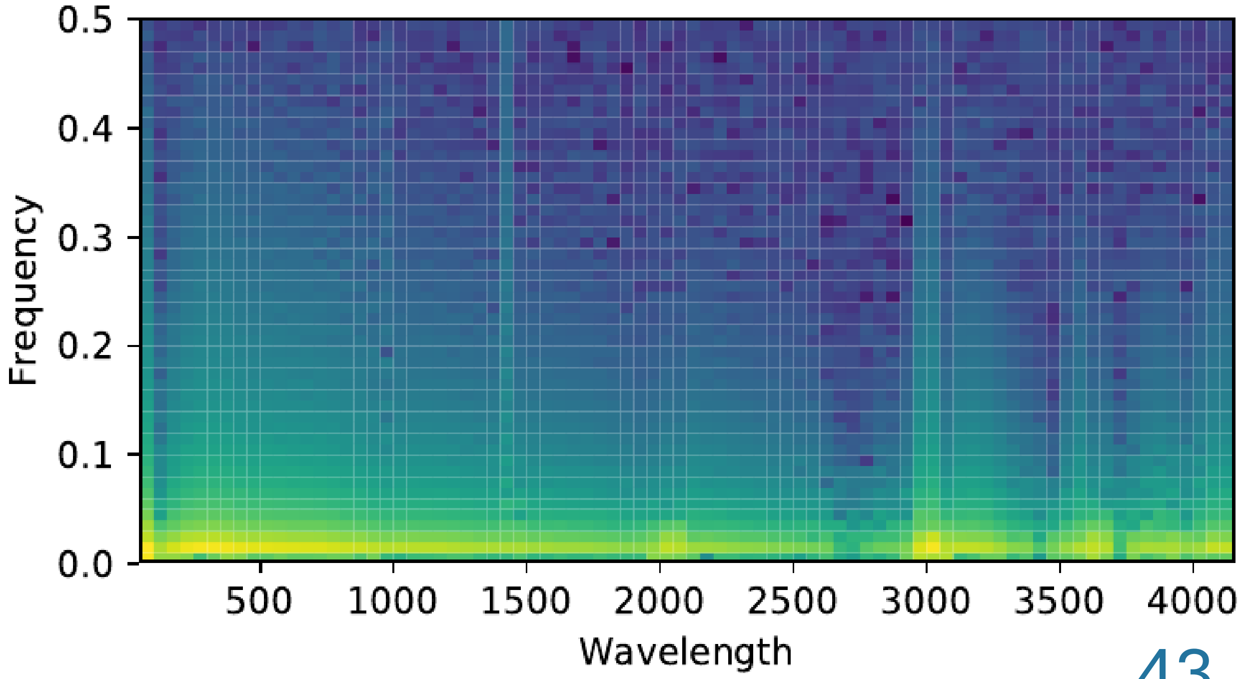
CNN 2D – lidando com espectrogramas

CNN 2D – lidando com espectrogramas

Como extrair todo o potencial da camada de convolução de uma CNN no contexto de dados espectrais 1D?

Uma possível saída é tratar os espectros como séries temporais e extrair a sua representação no domínio das frequências, gerando os espectrogramas (2D). Isso pode ser feito por meio da Transformada de Fourier de Tempo Curto





CNN 2D – lidando com espectrogramas

Uma série temporal é a representação de uma sequência de observações tomadas em intervalos de tempo periódicos. Ex: medida da temperatura em um determinado local, de hora em hora. No nosso caso o "tempo" é a energia/comp de onda

A ideia por traz da Transformada de Fourier (TF) é extrair as frequências básicas que geram os sinais incorporados nas séries temporais (domínio temporal: contagem x tempo) permitindo analisar estruturas invisíveis no domínio do tempo

Matematicamente, a TF decompõe qualquer sinal em uma soma de senos e cossenos, revelando o "ingrediente" de cada frequência presente. Na sua versão contínua a transformada é obtida via:

$$x(f) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi f t}dt = \int_{-\infty}^{\infty} x(t)[\cos(2\pi f t) - i sen(2\pi f t)]\, dt$$

CNN 2D – lidando com espectrogramas

$$x(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi f t} dt = \int_{-\infty}^{\infty} x(t)[\cos(2\pi f t) - i sen(2\pi f t)]\, dt$$

No contínuo, a TF integra $x(t)$ sobre todo t. No caso discreto, podemos discretizar o tempo em *N* intervalos Δt. Com amostragens em instantes $t_n = n\Delta t$ só teremos valores pontuais $x_n = x(t_n)$. Logo, para cada amostra x_n:

$$x(t) = \sum_{n=0}^{N-1} x_n \, \delta(t - n\Delta t)$$

$$x(f) = \sum_{n=0}^{N-1} x_n e^{-i2\pi f n\Delta t} \; ; n = 0, 1, \ldots, N-1$$

$$f \to f_k = \frac{k}{N\Delta t} \; ; \; k = 0, 1, \ldots, N-1$$

$$x(f_k) = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{kn}{N}}$$

CNN 2D – lidando com espectrogramas

$$x(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi f t} dt$$

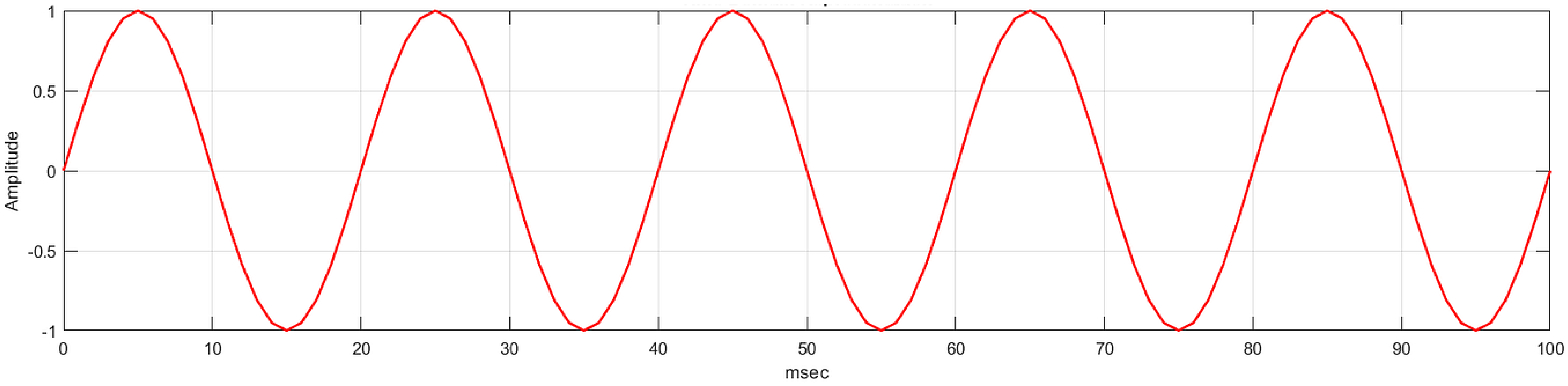
Caso contínuo

$$x(f_k) = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{kn}{N}}$$

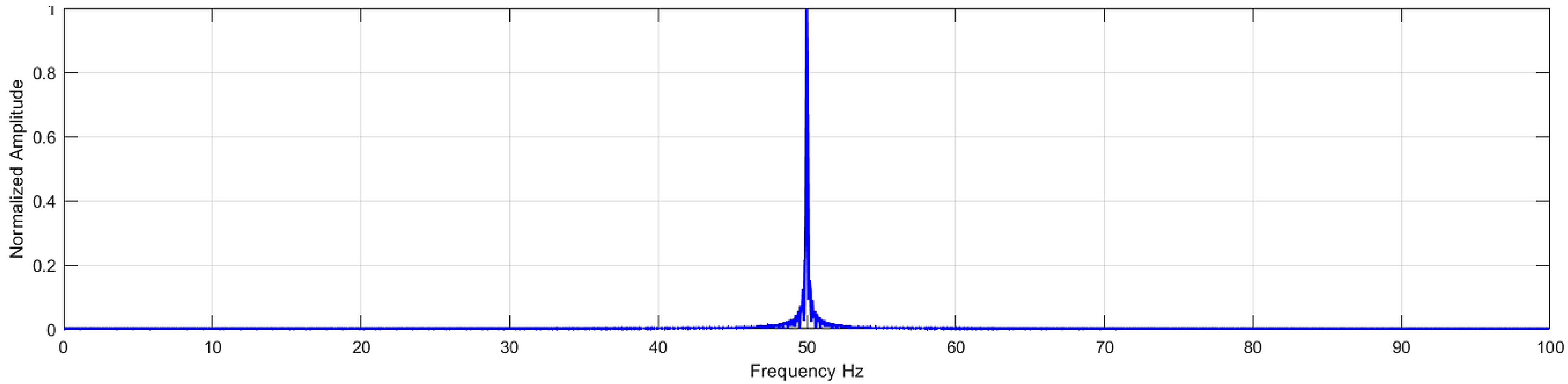
Caso discreto $t_n = n\Delta t, n, k = 0,1, \ldots N-1$

Como resultado, tanto $x(f)$ quanto $x(f_k)$ são as amplitudes no domínio de frequências (parte real + imaginária) e representam intensidades das contribuições de cada frequência nos sinais originais $x(t)$ e $x_n = x(t_n)$. Já o argumentos das amplitudes representam as frequências base dos sinais incorporadas na amplitudes originais (domínio temporal)

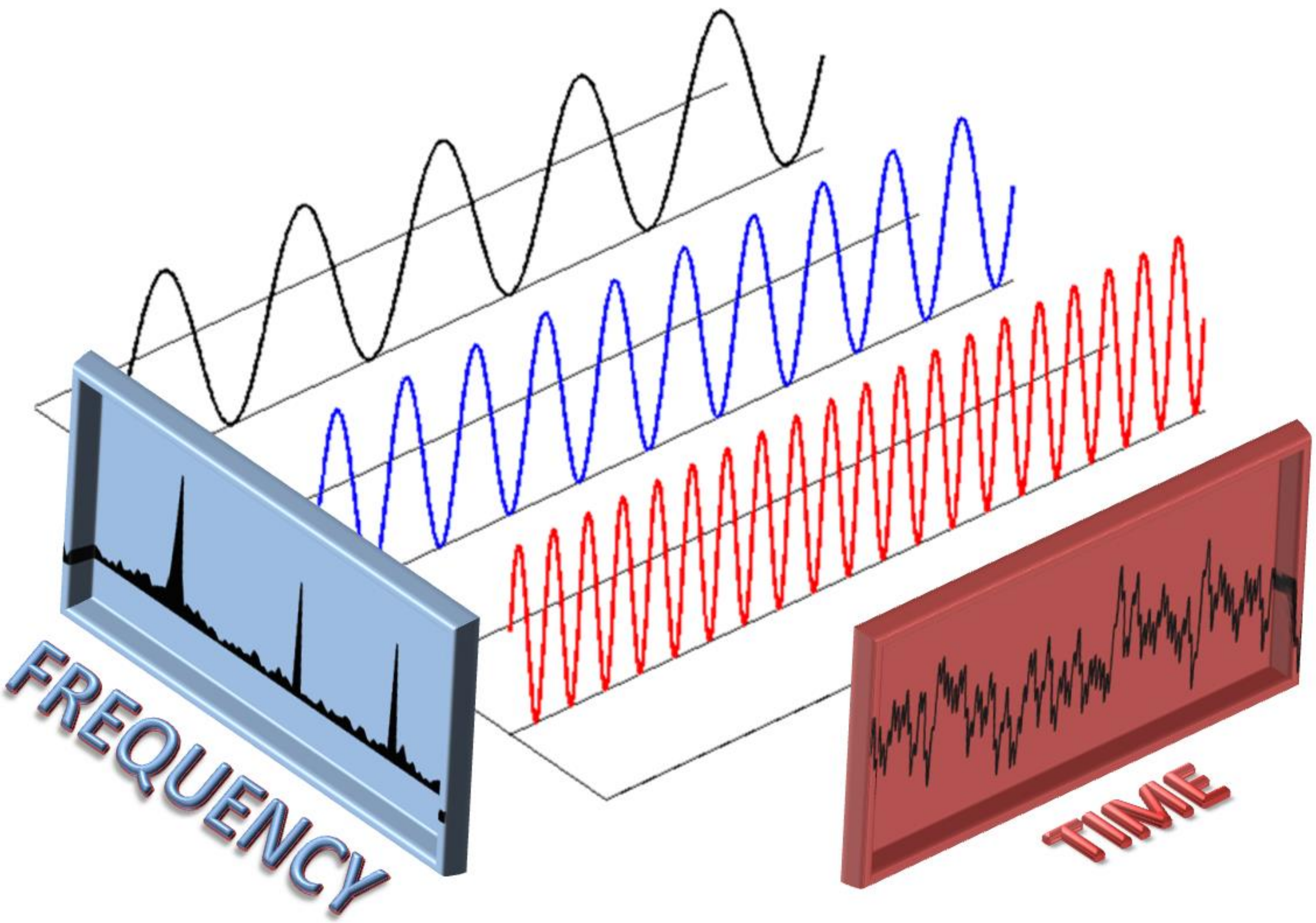
Domínio temporal (original)



Domínio das frequências

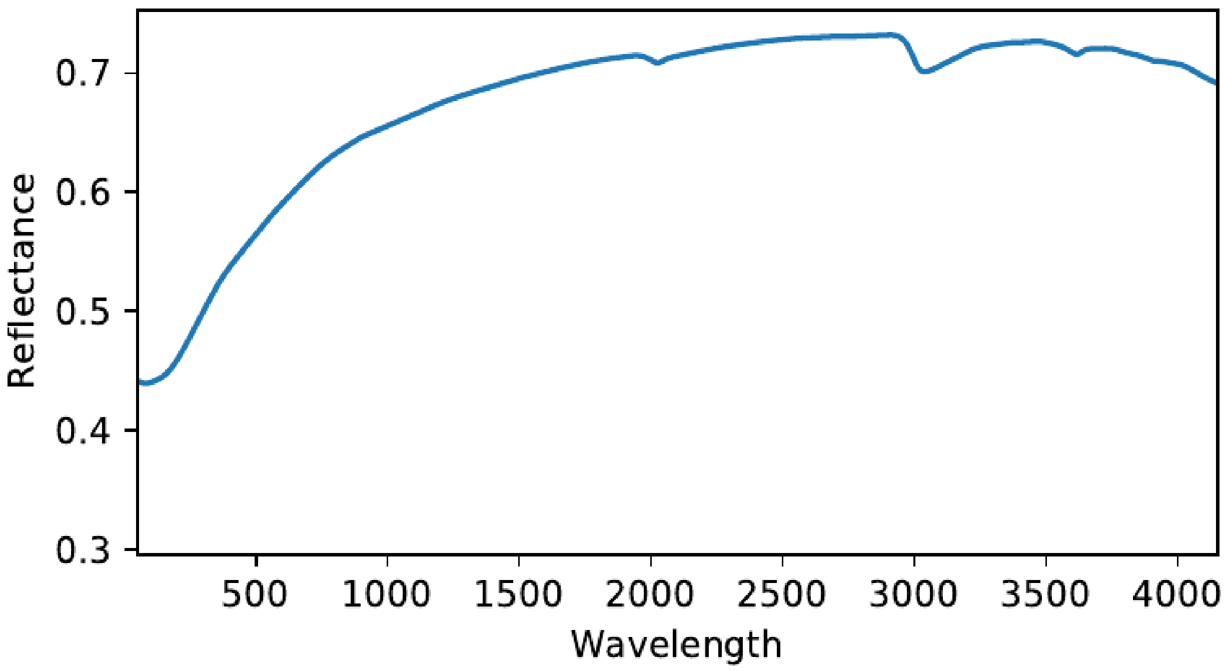


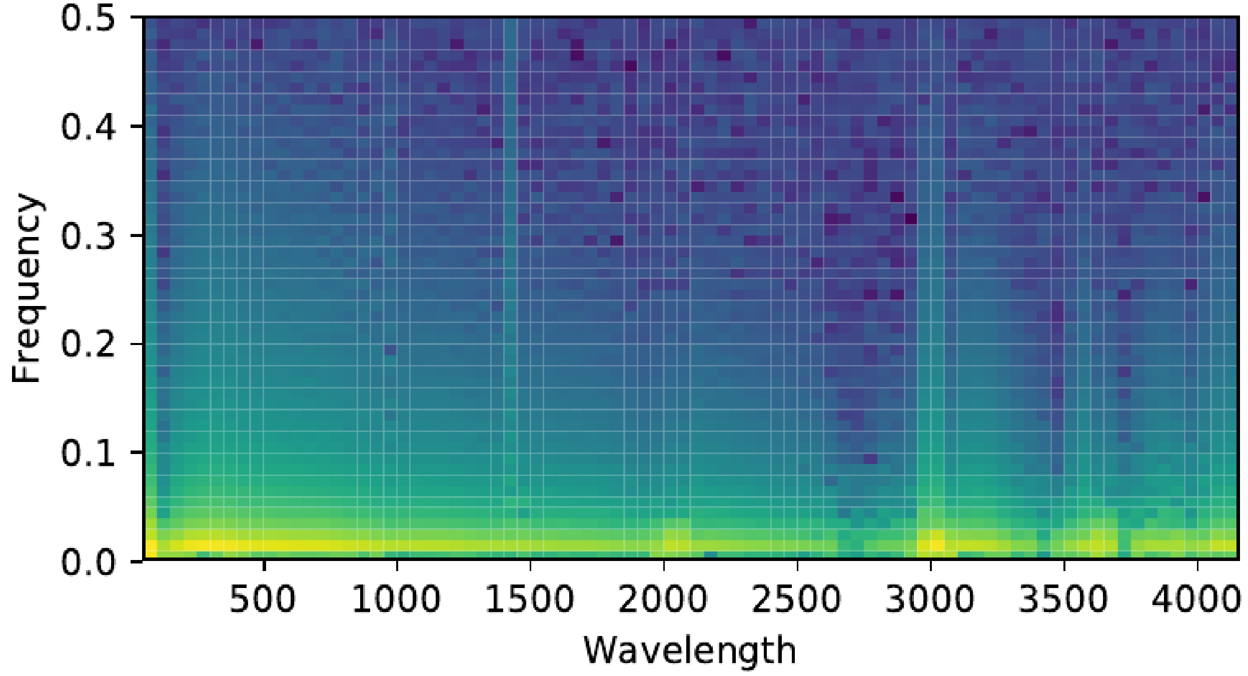
Domínio temporal (original) × Domínio das frequências



CNN 2D – lidando com espectrogramas

Espectros x Espectrogramas





Exemplo de estudos com vis-NIR:

10.1016/j.geodrs.2018.e00198

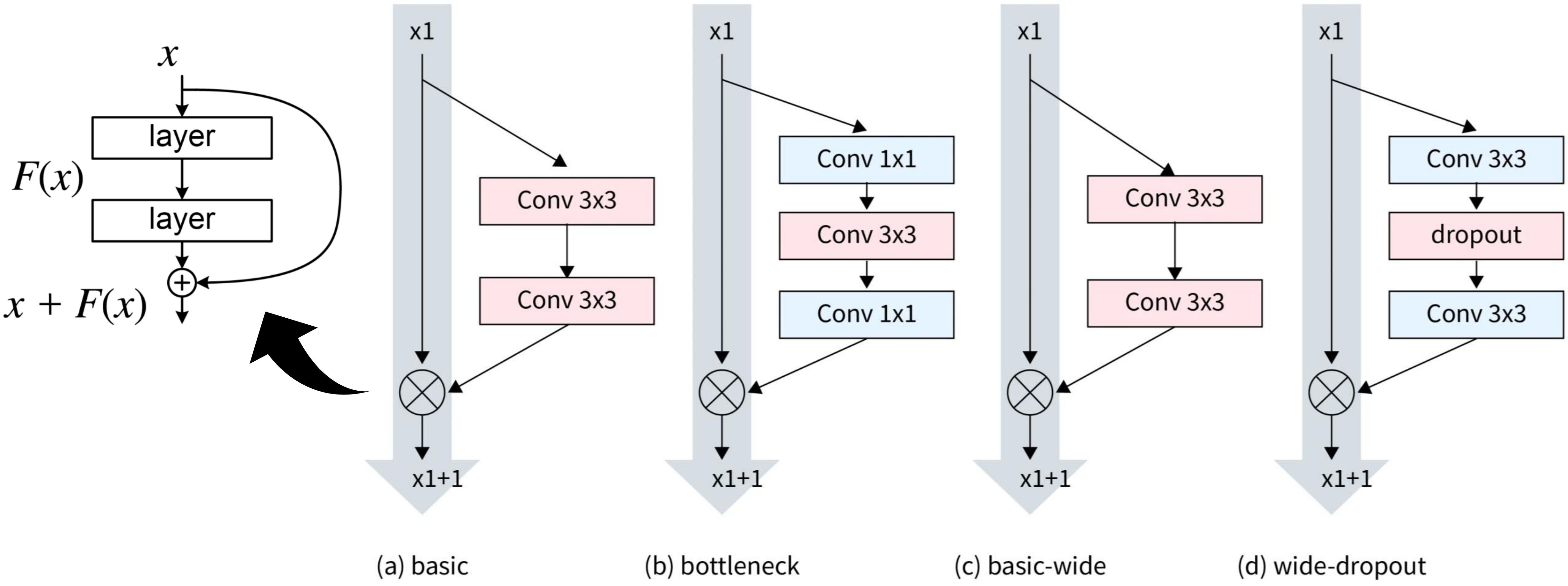
10.1016/j.geoderma.2019.01.009

10.3390/s20216271

Outras arquiteturas famosas

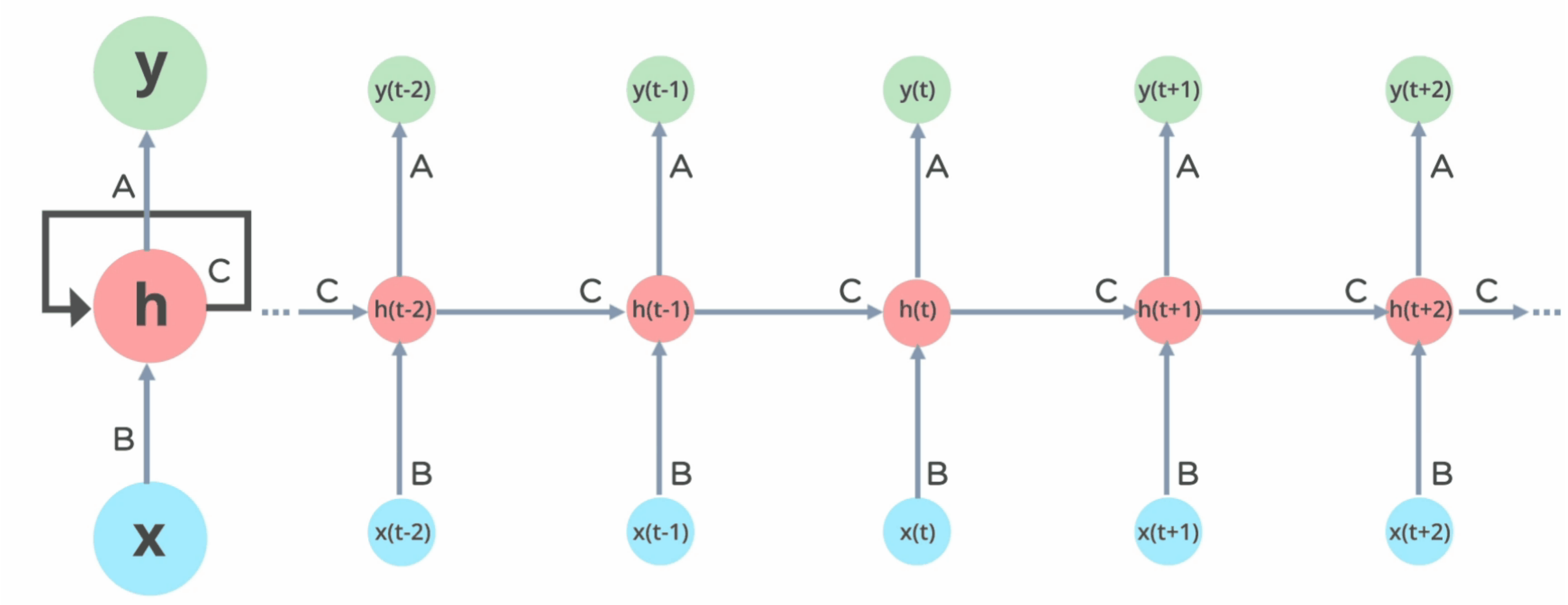
Outras arquiteturas famosas

Redes Residuais - ResNet



Outras arquiteturas famosas

Redes Recorrentes – RNN



Noções de *transfer learning*

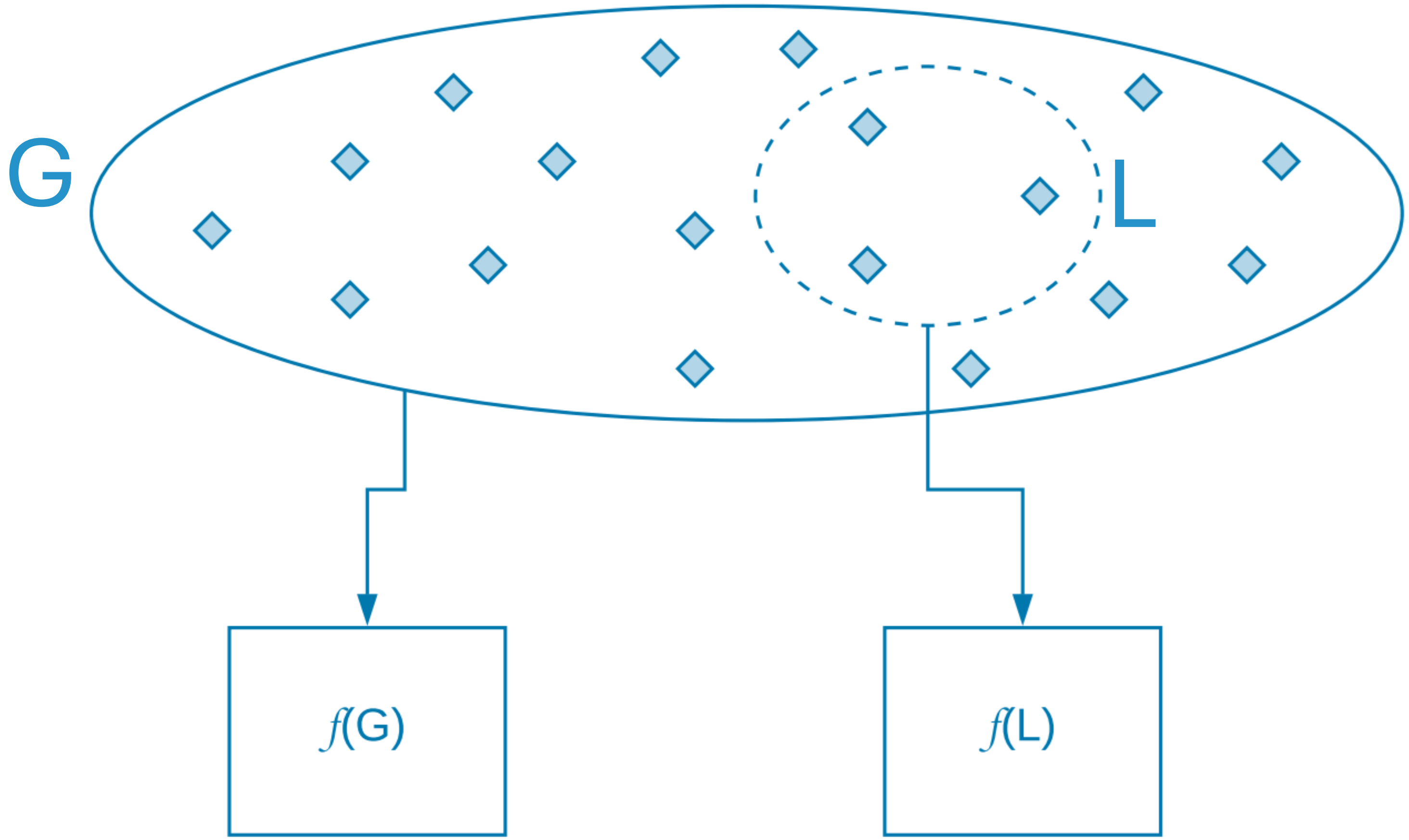
Noções de *transfer learning*

Humanos são capazes de aplicar conhecimento previamente adquirido a tarefas que têm características similares. O aprendizado por transferência (*transfer learning*), também conhecido como aprendizado por indução, é um ramo do ML que tenta emular esse processo

Em outras palavras, é uma abordagem na qual um modelo treinado para uma tarefa é reutilizado, ou ajustado, para uma nova tarefa relacionada, aproveitando o conhecimento adquirido em um domínio de origem para melhorar o desempenho em um domínio de destino, especialmente quando há dados limitados para a nova tarefa.

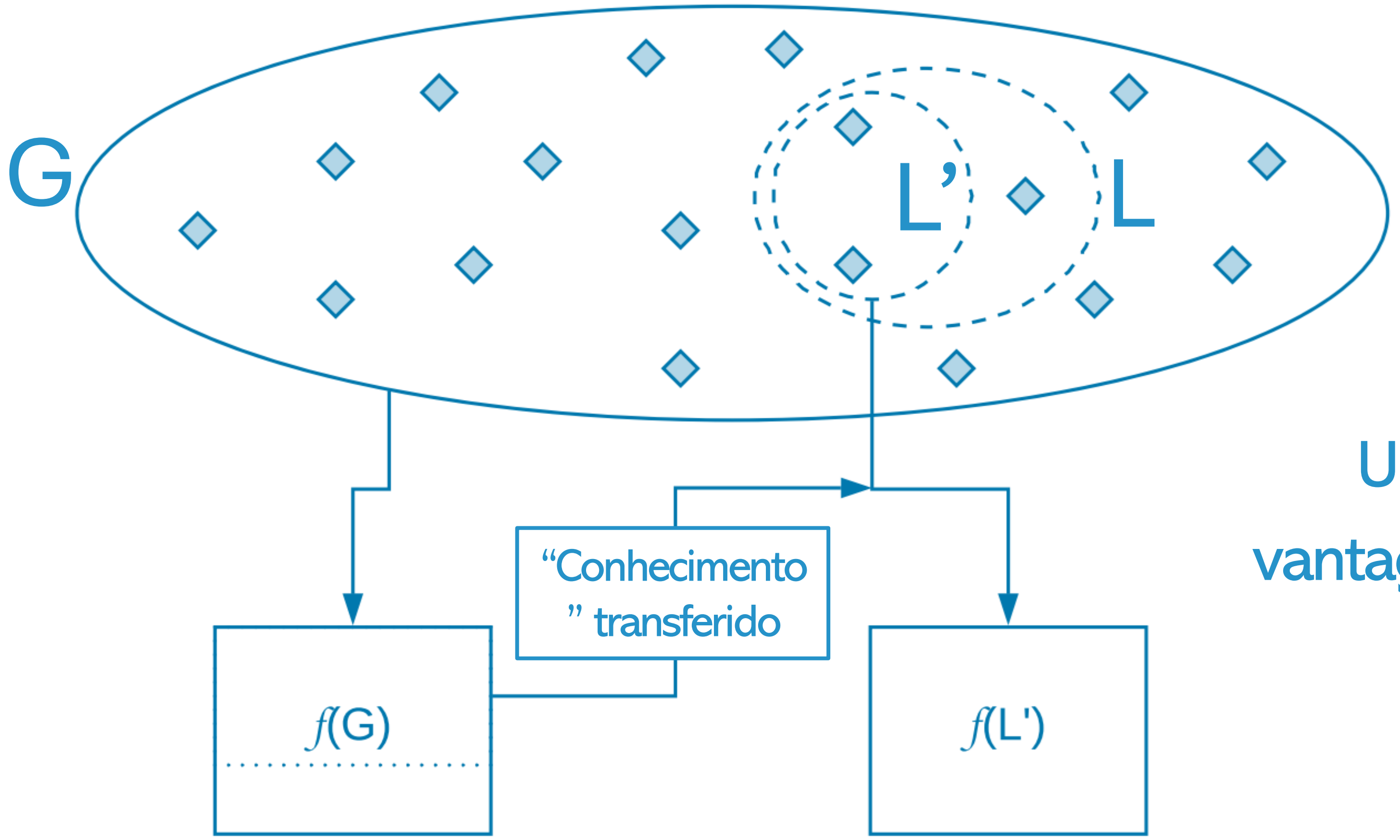
Noções de *transfer learning*

Dado um domínio de dados global **G** e um domínio de dados local **L** ($L \subset G$), uma abordagem tradicional de ML considera ambos os domínios como diferentes, gerando dois modelos independentes, $f(G)$ e $f(L)$



Noções de *transfer learning*

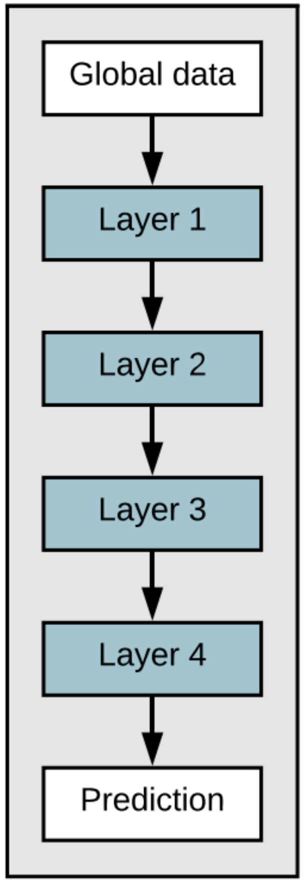
Reconhecendo que G e L estão de alguma forma relacionados, o *transfer learning* é capaz de gerar um modelo $f(L')$ usando parte das generalizações aprendidas por $f(G)$ em conjunto com o domínio de dados L', com $L' \subset L$



Uma das grandes vantagens é que $L' \ll L$

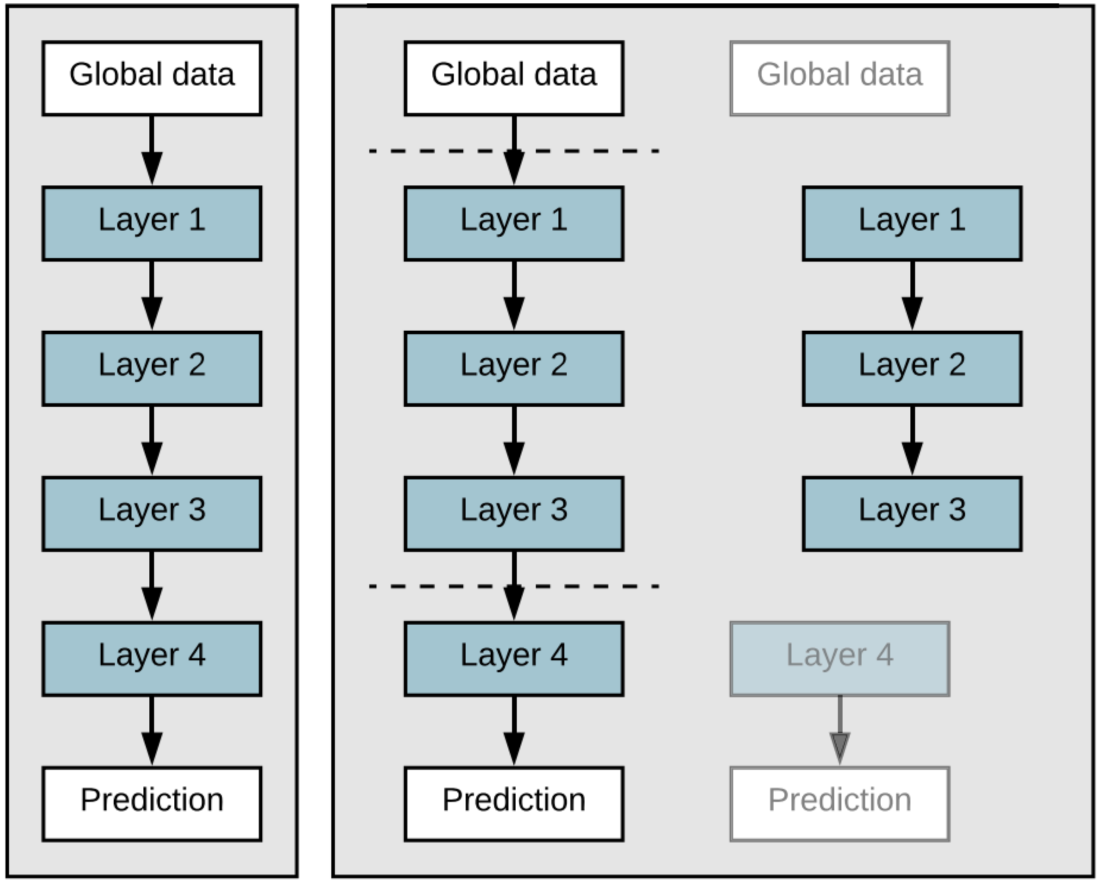
Noções de *transfer learning*

Esse procedimento é particularmente facilitado quando empregando modelos baseados em redes neurais, devido a versatilidade da arquitetura adotada



No primeiro treinamento (conjunto de dados global), o modelo gera uma representação interna da natureza global dos dados. Para aprender essa representação com sucesso, o modelo precisa de um grande volume de observações, que é exatamente o que o conjunto de dados global fornece

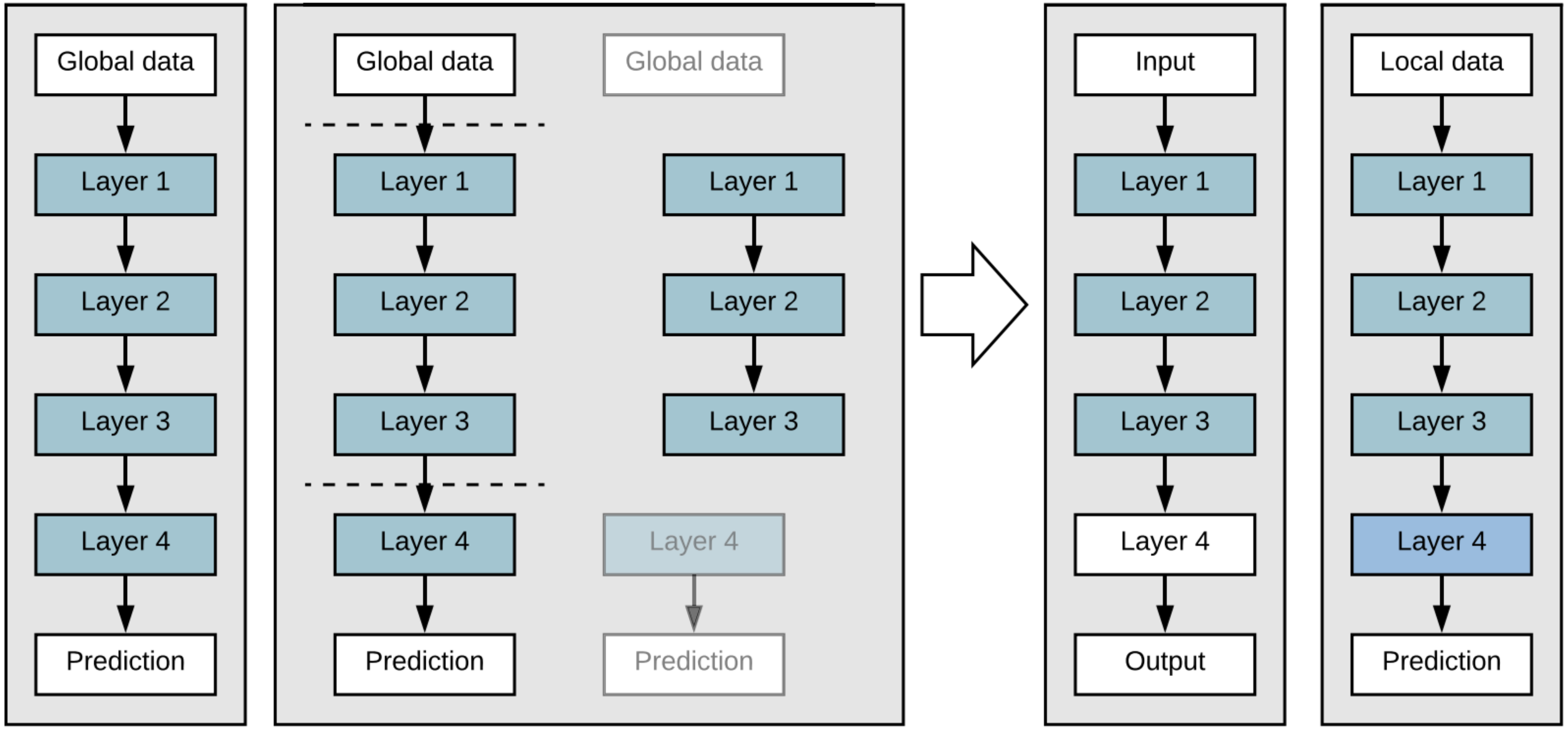
Noções de *transfer learning*



Subsequentemente, os parâmetros mais gerais, que aprenderam como os dados se comportam globalmente, podem ser extraídos

Noções de *transfer learning*

Tais parâmetros são então adaptado através de um número de novas observações que seja suficiente para ajustá-lo às condições locais desejadas



PRÁTICA – GOOGLE COLAB