Problemas Clássicos da Computação

Redes Neurais Convolucionais

Felipe Augusto Lima Reis

felipe.reis@ifmg.edu.br

Sumário

Introdução

Redes Neurais Convolucionais

INSTITUTO FEDERAL
Minas Gerais

"As redes convolucionais são simplesmente redes neurais que usam convolução no lugar da multiplicação geral da matriz, em pelo menos uma de suas camadas"
[Goodfellow et al., 2016].

Redes Neurais Convolucionais

INSTITUTO FEDERAL
Minas Gerais

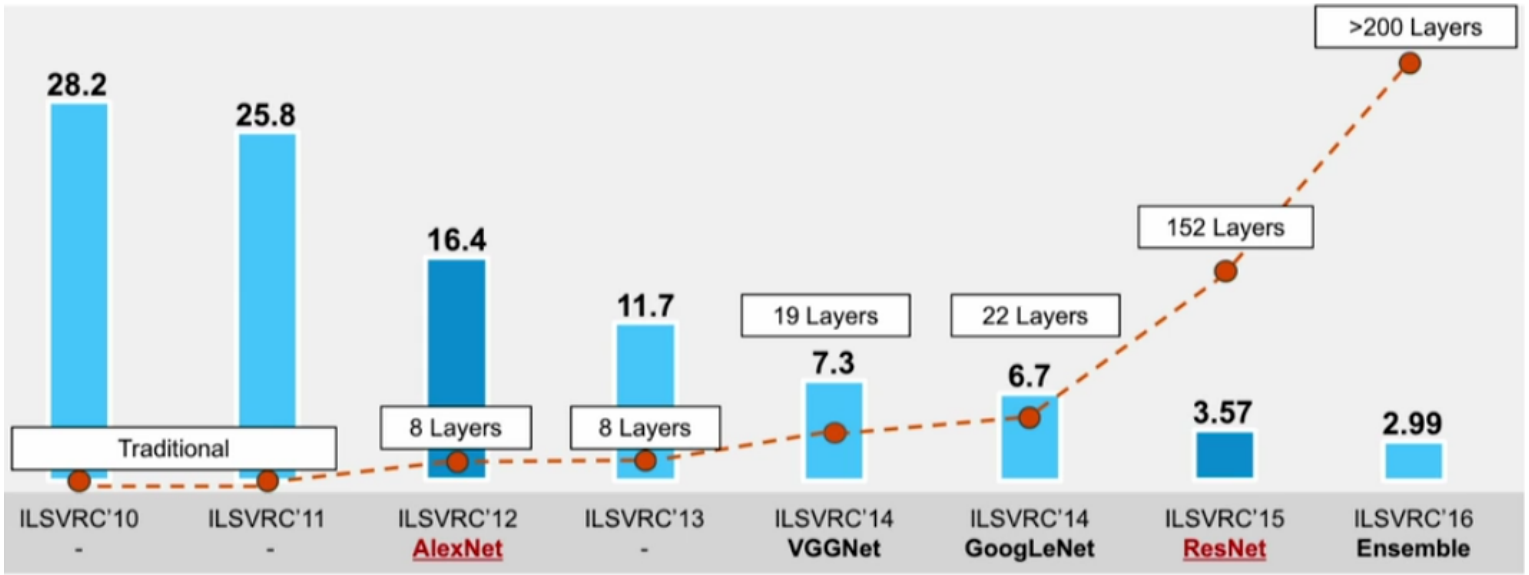
- Baseadas em uma teoria "antiga", ganhou protagonismo a partir da rede AlexNet, em 2012 [Florindo, 2018];
- A rede proposta por Alex Krizhevsky reduziu erro de 26% para 15% no ILSVRC[1] 2012[2] [Goodfellow et al., 2016] [Krizhevsky et al., 2012];
- Redes convolucionais permitiram o surgimento de diversas aplicações, principalmente na área de visão computacional.

[1] ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

[2] O erro é cálculado com bases nas 5 classes preditas pelos algoritmos.

Redes Neurais Convolucionais

- A partir das rede AlexNet, outras redes convolucionais estão relacionadas ao estado da arte em diversas aplicações;
- O número de camadas dessas redes também têm aumentado ao longo do tempo, conforme imagem abaixo.



Fonte: [Sadek Alaoui - SQLML, 2017]

Inspiração Biológica

INSTITUTO FEDERAL
Minas Gerais

- Podemos reconhecer objetos partindo de primitivas básicas e pequenas partes
  - Primitivas são características básicas como cores, diferenças de tonalidades, etc.;
  - Partes podem ser definidas como partes de objetos e/ou seres.
- Após o reconhecimento de características básicas e partes de objetos, podemos, iterativamente, construir conceitos mais complexos e/ou abstratos [Florindo, 2018].

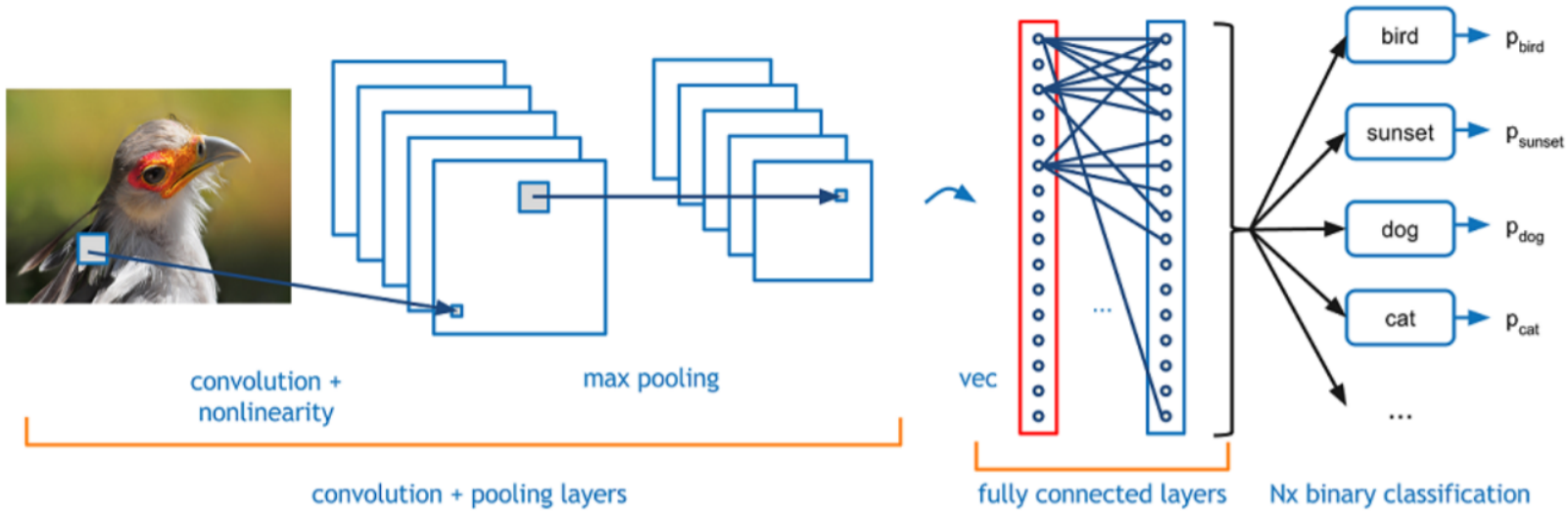
Inspiração Biológica

Fonte: Adaptado de [Wikipedia contributors (Tony Hisgett), 2020], [Wikipedia contributors (Harrie Gielen), 2020], [Wikipedia contributors (Charles J Sharp), 2020] e [Wikipedia contributors (Dominique Jacholke), 2020]

Arquitetura de Redes Convolucionais

Arquitetura de Redes Convolucionais

- Redes convolucionais (CNNs) são compostas por camadas com convoluções, operações de *pooling* e funções de ativação ReLU (nas camadas intermediárias);
- Dependendo do objetivo da rede, podem ser utilizadas camadas totalmente conectadas no final da rede [Li et al., 2021].



Fonte: [Florindo, 2018]

Arquitetura de Redes Convolucionais

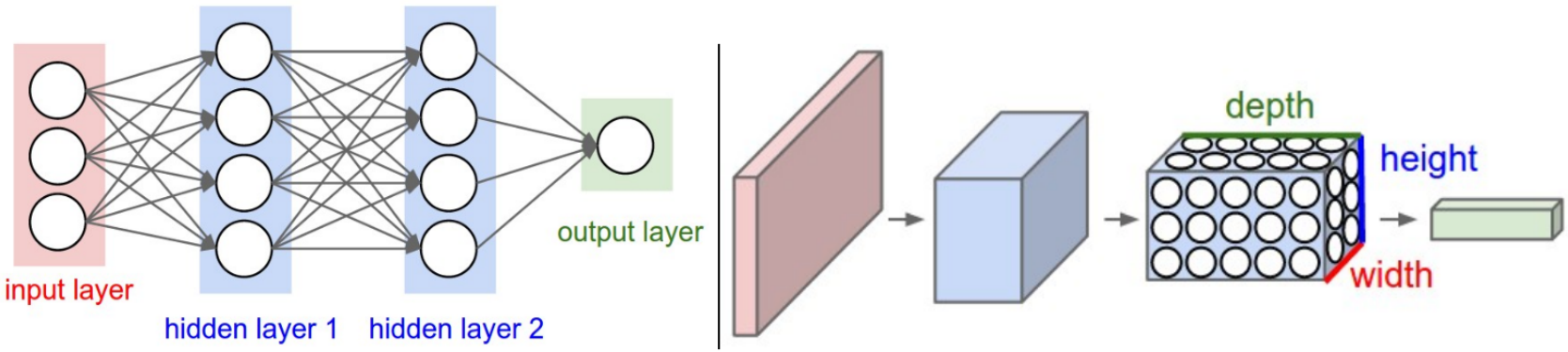
INSTITUTO FEDERAL
Minas Gerais

- Em geral, as redes convolucionais trabalham com imagens
  - Com isso, a arquitetura pode conter algumas restrições não existentes em redes tradicionais;
- Ao contrário de redes convencionais, redes convolucionais possuem neurônios organizados em 3 dimensões:
  - As dimensões são: largura, altura e profundidade[3];
  - Essa característica é especialmente útil no processamento de imagens;
  - Imagens possuem altura, largura e, em geral, 3 canais de cores [Li et al., 2021].

[3] Tradução do inglês: *width*, *height* e *depth*.

Arquitetura de Redes Convolucionais

- Ao contrário das redes totalmente conectadas:
  - Os neurônios em uma camada são conectados apenas a uma pequena região da camada anterior;
  - A rede transforma a entrada 3D em um "volume" 3D de ativações de neurônios;
  - No final de algumas redes existem camadas de achatamento (*flatten*) e/ou camadas totalmente conectadas, para produzir predições [Li et al., 2021].



Fonte: [Li et al., 2021]

Convolução

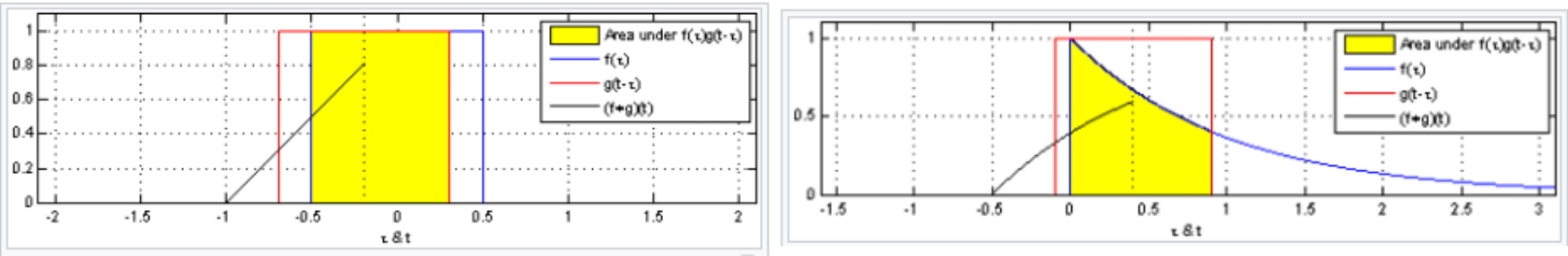
Operação de Convolução

- A convolução é a operação mais importante em uma rede convolucional;
- Consiste em um operador linear para duas funções f e g;
- Resulta em um terceiro valor s, que mede a soma do produto ao longo da região subentendida pela superposição de g, quando deslocada sobre uma função f [Weisstein, 2018].

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

Operação de Convolução

- A convolução pode ser entendida como a forma com que um sistema opera sobre um sinal de entrada [Smith, 1997].



Fonte: Adaptado de [Wikipedia contributors, 2020a]

Link: animação de uma convolução 1 [Wikipedia contributors, 2020a]

Link: animação de uma convolução 2 [Wikipedia contributors, 2020a]

Operação de Convolução

- Na terminologia de redes convolucionais, a função f corresponde a entrada, g representa o *kernel* e a saída s corresponde ao mapa de saída (*feature map*);
- Segundo [Goodfellow et al., 2016], as convoluções são utilizadas para aprimorar resultados nos seguintes campos:
  - Interações esparsas: possibilitam o armazenamento de características importantes com baixo número de parâmetros;
  - Compartilhamento de parâmetros: possibilita armazenamento apenas de alguns valores, que geram os demais pesos da rede;
  - Representações equivariantes: alterações realizadas na entrada produzem efeitos correspondentes nas saídas.

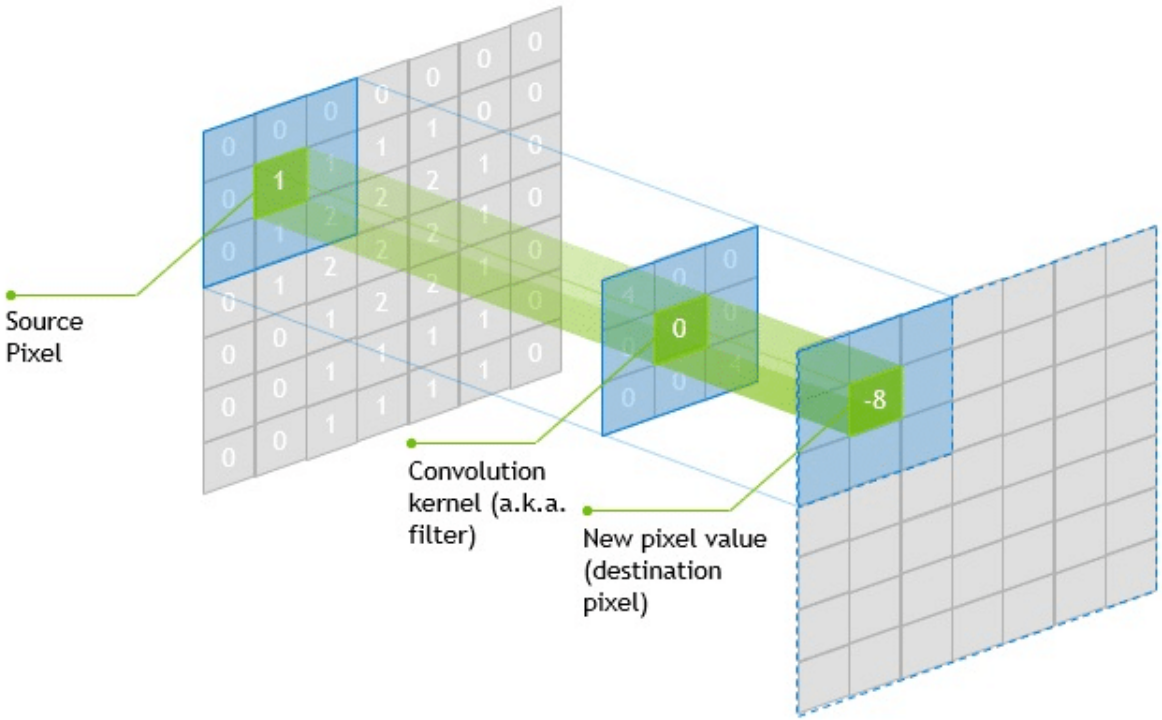
Camada Convolucional

- Principal camada das redes convolucionais, responsável pela maior parte dos cálculos [Li et al., 2021];
- Convoluções utilizam múltiplos filtros (*kernels*), que podem ter diferentes tamanhos (ex.: $5 \times 5 \times 3$, $32 \times 32 \times 3$, etc);
- Durante o *forward pass*, é feita a convolução de cada filtro na largura e altura do volume de entrada;
- Em seguida, são calculados os produtos escalares entre as entradas do filtro e a entrada em qualquer posição [Li et al., 2021].

Camada Convolucional

INSTITUTO FEDERAL
Minas Gerais

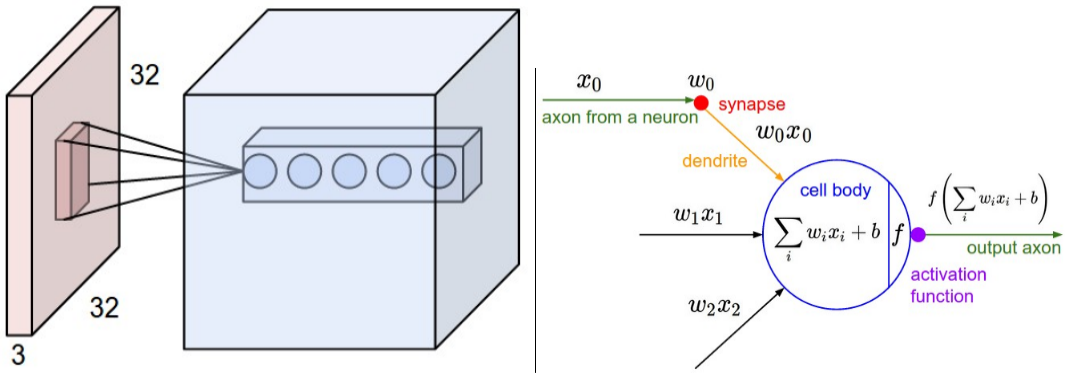
- Os filtros são "deslizados" sobre largura e altura do volume;



Fonte: [Martin, 2018]

Camada Convolucional

- Produz-se um mapa de ativação bidimensional que fornece as respostas desse filtro em cada posição espacial;
- A rede aprende quais filtros ativar a partir das entradas;
- Os filtros ativados são empilhados na camada de profundidade para produção do volume de saída [Li et al., 2021].

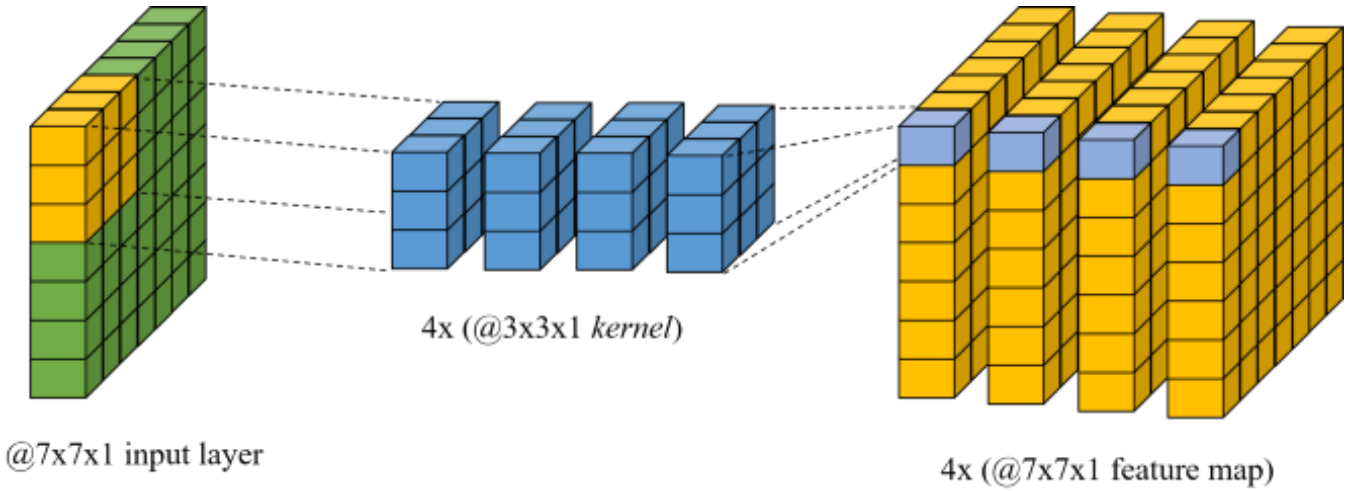


Fonte: [Li et al., 2021]

Link: Animação com o funcionamento de uma convolução [Li et al., 2021].

Camada Convolucional

- A aplicação de múltiplos filtros (*kernels*) convolucionais dá origem a uma profundidade diferente na camada seguinte [Li et al., 2021].



Fonte: [Yunus, 2020]

Link: Animação com o funcionamento de uma convolução [Li et al., 2021].

Profundidade, *Stride* e *Zero-padding*

INSTITUTO FEDERAL
Minas Gerais

- O tamanho do volume, é definido por 3 parâmetros:
  - Profundidade (*depth*): hiperparâmetro correspondente ao número de *kernels* que serão utilizados para criação de mapas de características
    - Cada filtro pode aprender características diferentes, como cores, texturas e variações de tonalidade;
  - *Stride*: taxa de "deslizamento" dos *kernels*, em *pixels*;
  - *Zero-padding*[4]: corresponde ao à quantidade de zeros adicionados às bordas da imagem para melhor ajuste de tamanho (*pad*) [Li et al., 2021] [IBM, 2020].

[4] Frequentemente denominado apenas como *padding*, pode ser traduzido como preenchimento de zeros.

Profundidade, *Stride* e *Zero-padding*

INSTITUTO FEDERAL
Minas Gerais

- O tamanho volume (largura e altura) de saída é dado por:

$$V = \frac{W - F + 2P}{S} + 1$$

onde

- W: tamanho do volume de entrada;
- F: tamanho do campo receptivo[5] do *kernel* na camada convolucional;
- S: *stride*;
- P: *zero-padding*;
- K: profundidade da convolução[6].

- O volume deve consistir em um número inteiro positivo;
  - Dependendo dos parâmetros, essa condição nem sempre é possível - os parâmetros devem, então, ser alterados.

[5] Do inglês *receptive field* (RF), correspondente ao tamanho da região na entrada que produz a *feature*.

[6] Não faz parte da fórmula de cálculo, mas influencia na profundidade do volume V.

Profundidade, *Stride* e *Zero-padding*

INSTITUTO FEDERAL
Minas Gerais

- Exemplo 1 ([Li et al., 2021])
  - Suponha o cálculo de um volume com os seguintes parâmetros: entrada $W = 10$, sem *padding* ($P = 0$) e filtro $F = 3$. O uso de um valor de *stride* $S = 2$ é possível?
    - Pela fórmula, temos:

$$V = \frac{W - F + 2P}{S} + 1 = \frac{10 - 3 + 0}{2} + 1 = 4.5$$

    - Não é possível, pois V não é um número inteiro positivo.

Profundidade, *Stride* e *Zero-padding*

INSTITUTO FEDERAL
Minas Gerais

- Exemplo 1 ([Li et al., 2021])
  - Suponha o cálculo de um volume com os seguintes parâmetros: entrada $W = 10$, sem *padding* ($P = 0$) e filtro $F = 3$. O uso de um valor de *stride* $S = 2$ é possível?
    - Pela fórmula, temos:

$$V = \frac{W - F + 2P}{S} + 1 = \frac{10 - 3 + 0}{2} + 1 = 4.5$$

    - Não é possível, pois V não é um número inteiro positivo.

Profundidade, *Stride* e *Zero-padding*

INSTITUTO FEDERAL Minas Gerais

- Exemplo 1 ([Li et al., 2021])
  - Suponha o cálculo de um volume com os seguintes parâmetros: entrada $W = 10$, sem *padding* ($P = 0$) e filtro $F = 3$. O uso de um valor de *stride* $S = 2$ é possível?
    - Pela fórmula, temos:

$$V = \frac{W - F + 2P}{S} + 1 = \frac{10 - 3 + 0}{2} + 1 = 4.5$$

    - Não é possível, pois V não é um número inteiro positivo.

Profundidade, *Stride* e *Zero-padding*

INSTITUTO FEDERAL
Minas Gerais

- Exemplo 2 (Adaptado de [Li et al., 2021])
  - A rede AlexNet[7], em sua primeira camada, utiliza os seguintes parâmetros: $W = 227$, $P = 0$, $F = 11$, $S = 4$ e $K = 96$. Qual o tamanho do volume de saída da camada?
    - A entrada do volume é definida, então, por $[227 \times 227 \times 3]$;
    - Pela fórmula, temos:

$$V = \frac{W - F + 2P}{S} + 1 = \frac{227 - 11 + 0}{4} + 1 = 54 + 1 = 55$$

    - A saída do volume possui o seguinte tamanho: $[55 \times 55 \times 96]$.

[7] ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky et al., 2012].

Profundidade, *Stride* e *Zero-padding*

INSTITUTO FEDERAL
Minas Gerais

- Exemplo 2 (Adaptado de [Li et al., 2021])
  - A rede AlexNet[7], em sua primeira camada, utiliza os seguintes parâmetros: $W = 227$, $P = 0$, $F = 11$, $S = 4$ e $K = 96$. Qual o tamanho do volume de saída da camada?
    - A entrada do volume é definida, então, por $[227 \times 227 \times 3]$;
    - Pela fórmula, temos:

$$V = \frac{W - F + 2P}{S} + 1 = \frac{227 - 11 + 0}{4} + 1 = 54 + 1 = 55$$

    - A saída do volume possui o seguinte tamanho: $[55 \times 55 \times 96]$.

[7] ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky et al., 2012].

Profundidade, *Stride* e *Zero-padding*

INSTITUTO FEDERAL
Minas Gerais

- Exemplo 2 (Adaptado de [Li et al., 2021])
  - A rede AlexNet[7], em sua primeira camada, utiliza os seguintes parâmetros: $W = 227$, $P = 0$, $F = 11$, $S = 4$ e $K = 96$. Qual o tamanho do volume de saída da camada?
    - A entrada do volume é definida, então, por $[227 \times 227 \times 3]$;
    - Pela fórmula, temos:

$$V = \frac{W - F + 2P}{S} + 1 = \frac{227 - 11 + 0}{4} + 1 = 54 + 1 = 55$$

    - A saída do volume possui o seguinte tamanho: $[55 \times 55 \times 96]$.

[7] ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky et al., 2012].

Profundidade, *Stride* e *Zero-padding*

INSTITUTO FEDERAL
Minas Gerais

- Exemplo 2 (Adaptado de [Li et al., 2021])
  - A rede AlexNet[7], em sua primeira camada, utiliza os seguintes parâmetros: $W = 227$, $P = 0$, $F = 11$, $S = 4$ e $K = 96$. Qual o tamanho do volume de saída da camada?
    - A entrada do volume é definida, então, por $[227 \times 227 \times 3]$;
    - Pela fórmula, temos:

$$V = \frac{W - F + 2P}{S} + 1 = \frac{227 - 11 + 0}{4} + 1 = 54 + 1 = 55$$

    - A saída do volume possui o seguinte tamanho: $[55 \times 55 \times 96]$.

[7] ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky et al., 2012].

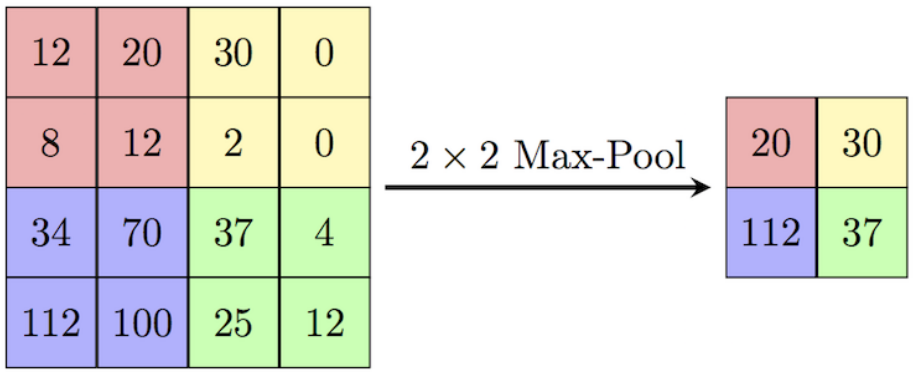
Tipos de *Zero-padding*

- *Zero-padding* é geralmente usado quando os filtros não se ajustam à imagem de entrada [IBM, 2020];
- Alguns *frameworks* conseguem ajustar o tamanho de forma automática, permitindo as seguintes opções:
  - *Valid padding*: não realiza *padding* (pode cortar elementos na última convolução);
  - *Same padding*: garante que a camada de saída tenha o mesmo tamanho que a camada de entrada;
  - *Full padding*: aumenta o tamanho da saída adicionando zeros à borda da entrada [IBM, 2020].

Pooling

Pooling

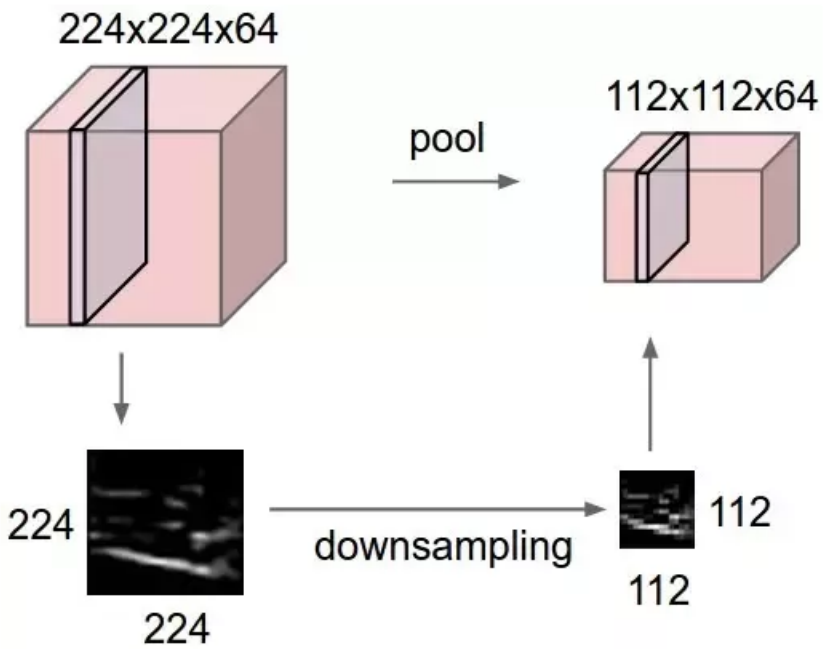
- A função de *pooling* é utilizada para prover informação estatística a respeito das saídas próximas;
- Tem como objetivo maximizar as vantagens da saída e tornar-se invariante a pequenos ruídos;
- Podem também ser utilizadas para redução da quantidade de neurônios entre camadas da rede, agrupando resultados na camada seguinte [Goodfellow et al., 2016].



Fonte: [Computer Science Wiki, 2020]

Pooling

- A operação de *pooling* é executada em cada camada do volume, promovendo redução de dimensionalidade;



Fonte: [Computer Science Wiki, 2020]

Pooling

- A operação de *pooling* está relacionada aos seguintes conceitos / características:
  - *Pool-size*[8]: Tamanho do filtro de *pooling* (ex.: 2*x*2, 3*x*3, etc);
  - *Stride*: taxa de "deslizamento" dos filtros, em *pixels*;
  - *Zero-padding*: quantidade de zeros adicionados às bordas da imagem para melhor ajuste de tamanho [IBM, 2020].

[8] Também chamado de *pooling size*.

Pooling

- Os tipos de pooling mais comuns são:
  - Max-pooling: utiliza operação de máximo no filtro de *pooling*
    - Tipo mais utilizado de *pooling*, presente em redes como a VGGNet e AlexNet;
    - Capaz de selecionar os *pixels* mais ativados localmente, descartando *pixels* com valores baixos;
    - Capaz de representar uma região, por meio das informações mais importantes;
  - Avg-pooling: utiliza a operação de média no filtro de *pooling*
    - Capaz de representar uma região, por meio de informações médias;
    - Menos comum que *max-pooling*, mas presente na rede ResNet.

Funções de Ativação

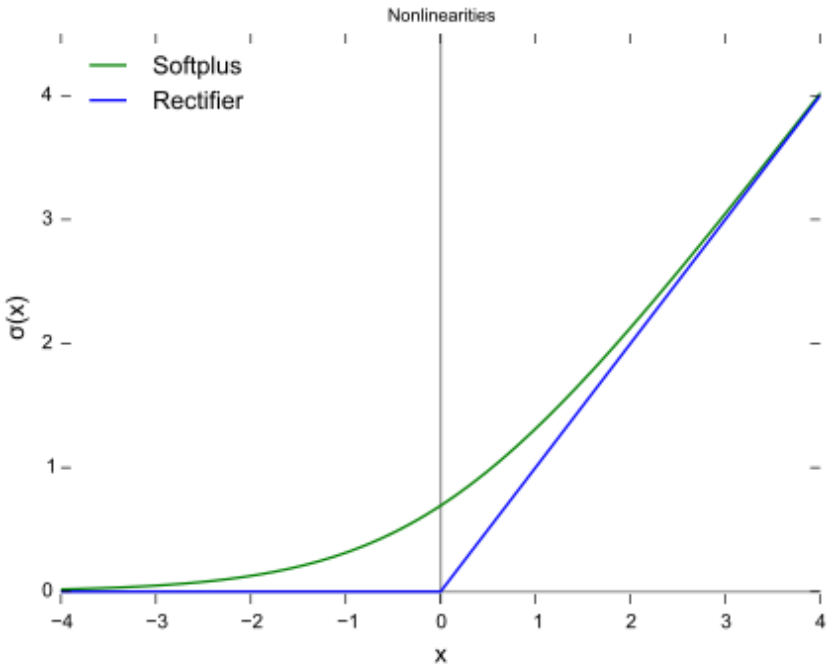
ReLU

- ReLU: *Rectified linear unit* (Unidade linear retificada);
- Função de ativação empregada em redes neurais convolucionais, principalmente em camadas intermediárias;
- Vantagens:
  - Melhor propagação de gradientes: diminuição da quantidade de gradientes que tendem a zero devido ao acúmulo de camadas e pequenos valores, quando comparado a função sigmoide;
  - Invariante a escala: $max(0, ax) = a\ max(0, x)$ para $a \geq 0$;
  - Ativação esparsa: em redes inicializadas aleatoriamente, somente 50% das camadas intermediárias são ativadas;

ReLU

- A função ReLU é definida formalmente como:

$$f(x) = max(0, x)$$



Fonte: [Wikipedia contributors, 2020b]

Variações da função ReLU

- Softplus: versão suavizada da função ReLU

$$f_{softplus}(x) = ln(1 + e^x)$$

- ELU: acrônimo de Exponential Linear Unit, possui um parâmetro α que pode ser ajustado
  - Potencialmente melhor que a função ReLU, porém mais cara computacionalmente;

$$f_{elu}(x) = \begin{cases} \alpha(e^x - 1) & \text{se } x \leq 0 \\ x & \text{se } x > 0 \end{cases}$$

Variações da função ReLU

- SELU: acrônimo de Scaled Exponential Linear Unit, corresponde a uma modificação da função ELU, para melhor acurácia e normalização.
  - Potencialmente ainda melhor que ELU, porém ainda mais cara;

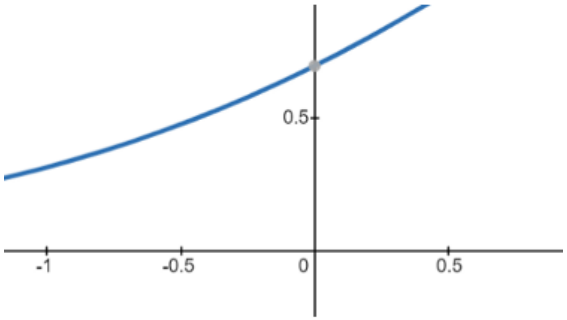
$$f_{selu}(x) = \lambda \times f_{elu}(x)$$

- Leaky ReLU: usada para suavizar o problema da função ReLU, que não permite gradientes negativos[9]
  - Suaviza o problema, multiplicando gradientes negativos por um valor pequeno.

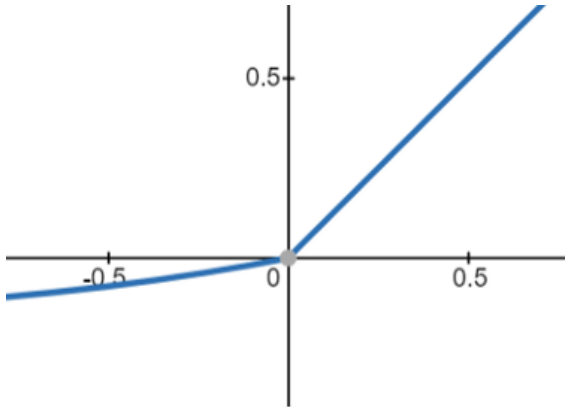
$$f_{lrelu}(x) = \begin{cases} 0.01x & \text{se } x < 0 \\ x & \text{se } x \geq 0 \end{cases}$$

[9] Problema conhecido como *Dying ReLU*.

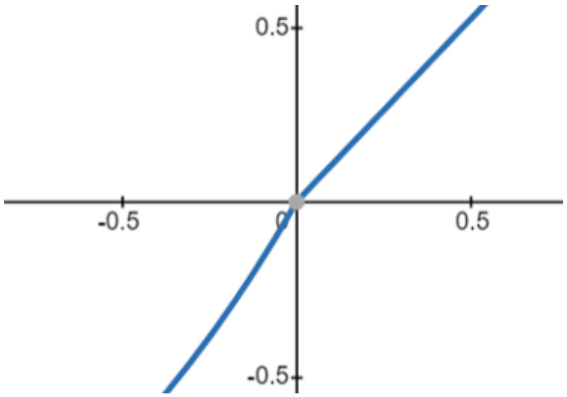
Variações da função ReLU



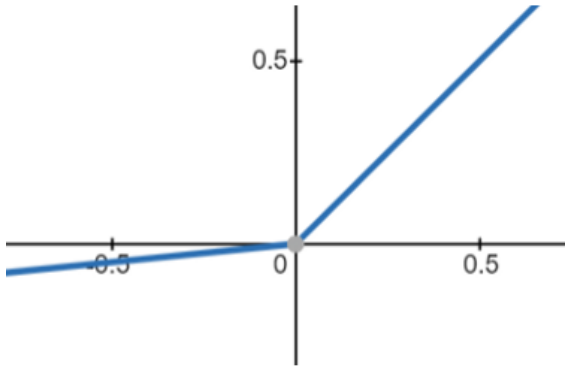
Softplus



ELU



SELU



Leaky ReLU

Fonte: [Chaudhary, 2020]

Otimizadores

Otimizadores

- Otimizadores correspondem aos métodos numéricos utilizados para minimização do erro esperado
  - O objetivo é reduzir a diferença entre a saída predita e a saída esperada;
  - Para isso, devemos minimizar a perda, definindo valores para os pesos da rede;
  - O método mais simples para isso é o *Gradient Descent (GD)*[10], porém esse método tem um custo muito alto
    - Dessa forma o *Gradient Descent* é muito pouco usado.

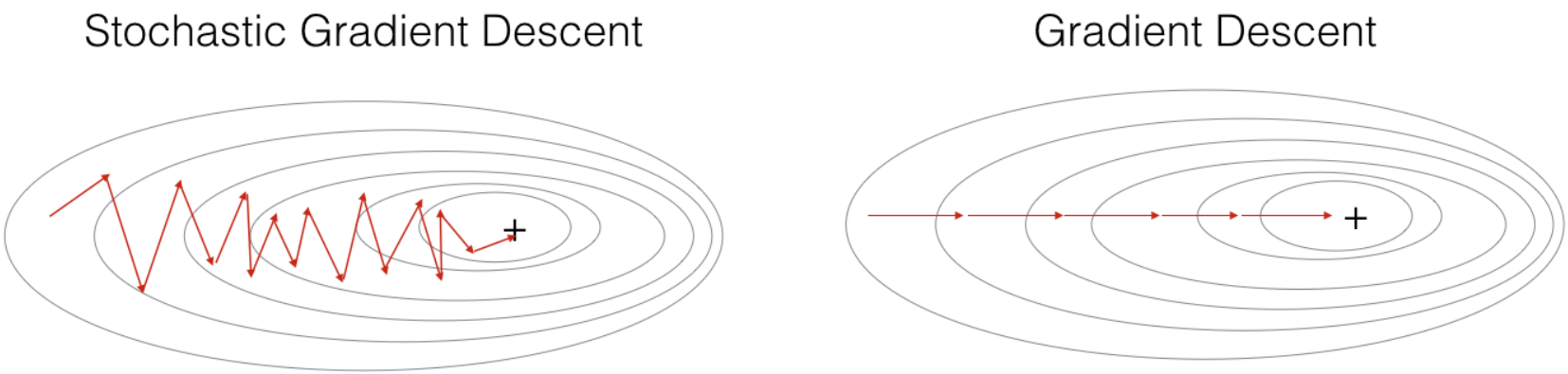
[10] Também chamado de *Batch Gradient Descent* (BDG) ou *Vanilla Gradient Descent* (VGD).

Stochastic Gradient Descent (SGD)

- O método do gradiente estocástico (SGD) corresponde a uma alteração no método *Gradient Descent*
  - O algoritmo não analisa todos os pontos a serem otimizados, apenas amostras dos dados;
  - O algoritmo varre as amostras e, calcula os valores de gradiente para cada delas;
  - O algoritmo toma a direção de maior declividade dentro das amostras;
  - Essa modificação reduz consideravelmente o custo computacional de cálculo dos gradientes [Ruder, 2016] [Hansen, 2019].

Stochastic Gradient Descent (SGD)

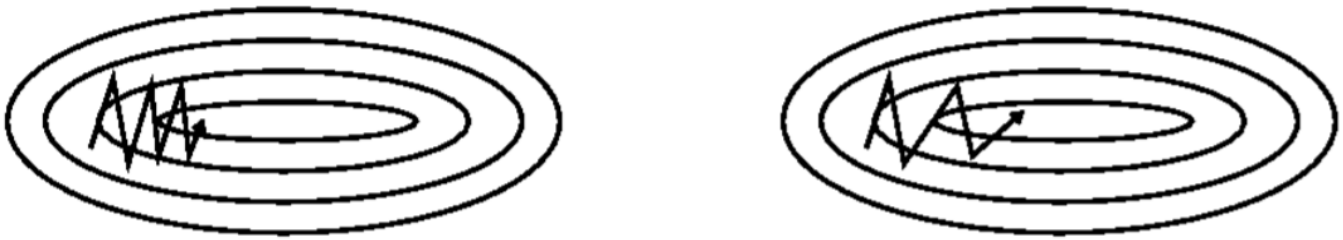
- Como o SGD analisa apenas amostras aleatórias dos dados, seu comportamento é mais instável que o algoritmo GD;
- No entanto, a redução de custo computacional compensa os efeitos colaterais [Ruder, 2016] [Yakout, 2021].



Fonte: [Yakout, 2021]

Momentum

- Momentum é um método para aceleração do SGD[11] quando as superfícies da curva convergem mais abruptamente em uma direção do que em outra;
- O Momentum adiciona um elemento temporal à equação;
- Seu funcionamento é análogo ao momento em física e auxilia o SGD a manter uma direção fixa [Ruder, 2016].



(a) SGD without momentum (b) SGD with momentum

Fonte: [Ruder, 2016]

[11] Momentum também pode ser utilizado com outros métodos de gradientes.
Link: animação de um SGD com momentum [Hansen, 2019].

Outros Otimizadores

INSTITUTO FEDERAL
Minas Gerais

- Além dos métodos previamente citados (mais simples), destacam-se também na literatura:
  - NAG: acrônimo de Nesterov Accelerated Gradient, tenta dar ao Momentum um tipo de presciência, de forma a dar saltos grandes no início da execução e evitando saltos à medida em que o algoritmo aproxima-se do objetivo [Ruder, 2016];
  - AdaGrad: adapta a taxa de aprendizado aos parâmetros, realizando atualizações maiores para parâmetros pouco frequentes e atualizações menores para parâmetros frequentes [Ruder, 2016].

Link 1: animação de otimizadores para redes neurais [Hoang Duong - Hoang Duong blog, 2015].

Link 2: animação de otimizadores para redes neurais [Ruder, 2016] [Radford, 2014].

Link 3: animação de otimizadores para redes neurais [Radford, 2014].

Outros Otimizadores

- Outros otimizadores:
  - Adadelta: melhoria do AdaGrad que busca reduzir o decrescimento agressivo e monotônico da taxa de aprendizado, ao reduzir o acúmulo de gradientes passados a uma janela de tempo fixa [Ruder, 2016];
  - RSMprop: método não publicado proposto por Geoffrey Hinton e semelhante ao Adadelta (desenvolvidos separadamente);
  - Adam: acrônimo de Adaptive Moment Estimation, consiste em um método adaptativo para ajuste das taxas de aprendizado, podendo ser considerado uma evolução dos métodos Adadelta/RMSprop e Momentum [Ruder, 2016].

Link 4: animação de otimizadores para redes neurais [Hansen, 2019] [Rahman, 2017].

Link 5: animação de otimizadores para redes neurais [Hansen, 2019] [Rahman, 2017].

Regularização

Regularização

- Regularização consiste em estratégias usadas para redução de erro no conjunto de teste às custas do aumento do erro de treinamento
  - Supõe-se que penalizar erros no treinamento e adicionar restrições extras ao problema podem aumentar a capacidade generalização dos algoritmos;
  - Qualquer modificação no aprendizado com o objetivo de reduzir seu erro de generalização, mas não seu erro de treinamento pode ser considerada uma regularização.

Regularização L_2

- A Regularização L_2 é popularmente conhecida como decaimento de pesos (*weight decay*)
  - Consiste na adição de um termo de regularização ω na função objetivo, de modo que a rede utilize todos os seus neurônios;
  - Evita que a rede use apenas neurônios que tiveram bons resultados em uma etapa inicial do treinamento;
- Penaliza vetores de pesos com valores altos e prefere vetores com pesos difusos,
  - Torna os pesos pequenos, mas não os zera, gera soluções não esparsas [Goodfellow et al., 2016] [Deshmukh, 2020].

Regularização L_2 também é conhecida como Ridge Regression.

Regularização L_1

- A Regularização L_1 é um tipo de regularização que leva o vetor de peso a ter valores próximos de zero durante a etapa de treinamento.
- Utiliza apenas um subconjunto distribuído no vetor de pesos (esparso) das entradas mais importantes, diminuindo a susceptibilidade às entradas que contém ruídos [Goodfellow et al., 2016].

Regularização L_2 também é conhecida como Lasso Regression.

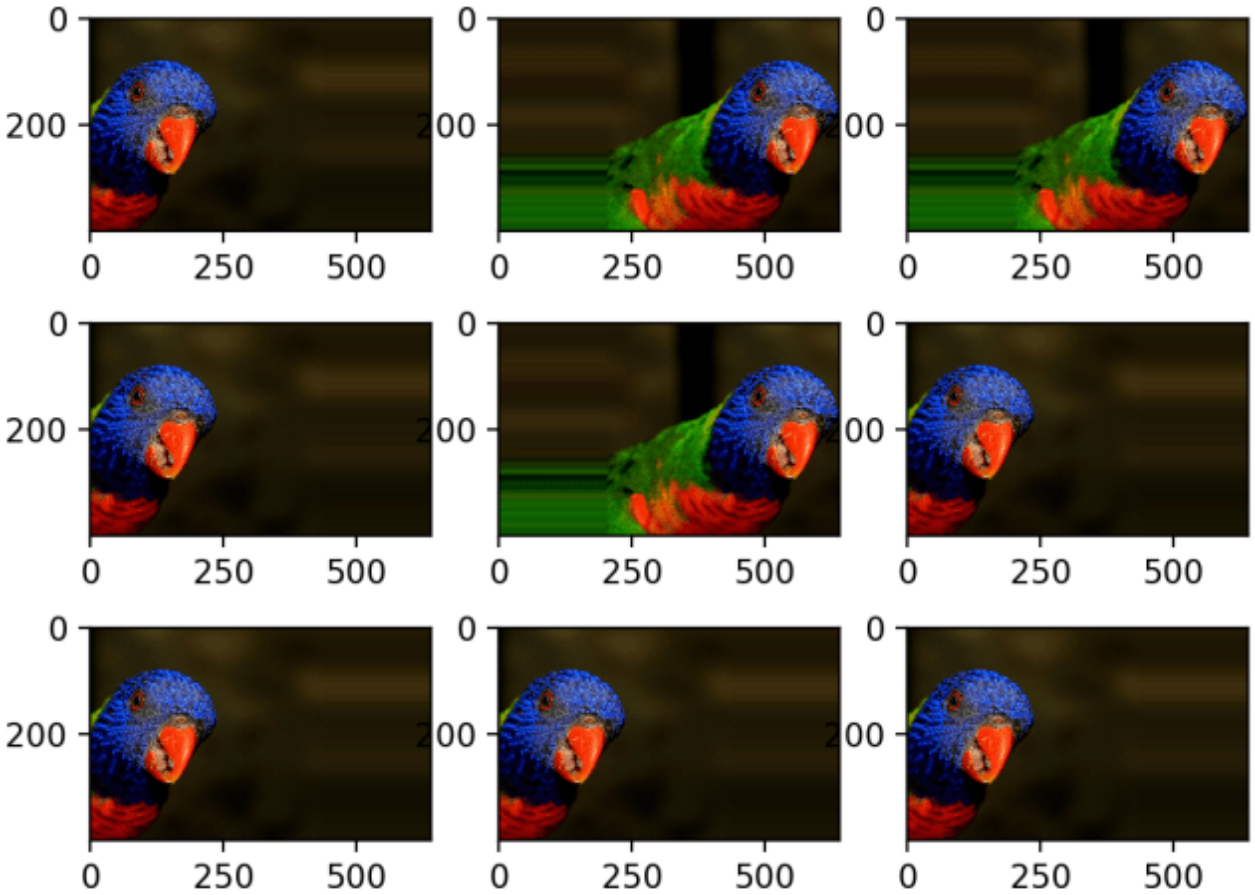
Data Augmentation

INSTITUTO FEDERAL
Minas Gerais

- Dataset augmentation[12] consiste em um conjunto de transformações aplicadas à base de dados para aumentar o número de registros para treinamento.
  - É aplicada principalmente a bases de dados com poucos registros, em modelos de treinamento não generalizam bem;
  - Este procedimento visa aumentar a precisão e robustez dos classificadores [Fawzi et al., 2016] [Perez and Wang, 2017];
  - Pode atuar como um tipo de regularizador evitando *overfitting* e aumentando o desempenho em problemas onde ocorre desequilíbrio de classes [Wong et al., 2016].

[12] Tradução direta: aumento do conjunto de dados.

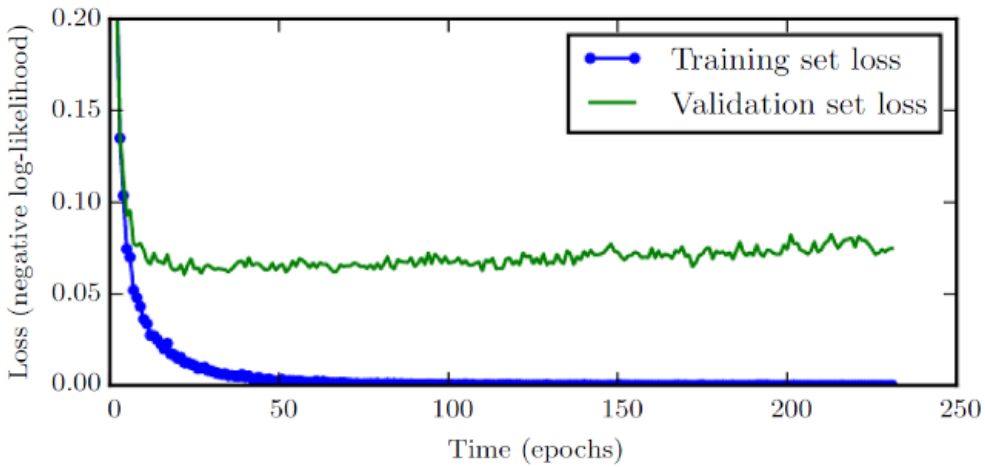
Data Augmentation



Fonte: [Deshmukh, 2020]

Early Stopping

- Early Stopping[13] corresponde à técnica de interrupção antecipada do treinamento
  - Após uma determinada quantidade de épocas, o erro de treinamento continua diminuindo, mas o erro de validação começa a aumentar, indicando possível *overfitting*;
  - Nesse momento, o treinamento pode ser interrompido para melhor desempenho [Deshmukh, 2020] [Upadhyay, 2019].



Fonte: [Upadhyay, 2019]

[13] Tradução literal: parada antecipada.

Bagging

- Bagging é uma técnica para redução do erro de generalização, por meio da criação de várias versões de um preditor;
  - Esse preditores são usados para criar um preditor agregado
  - Os valores de previsão são usados para um único resultado numérico baseado em votos individuais;
  - Técnicas que usam essa estratégia são chamados *ensemble methods* [Breiman, 1996] [Goodfellow et al., 2016].

Bagging é a abreviatura de *bootstrap aggregating*.

Ensemble

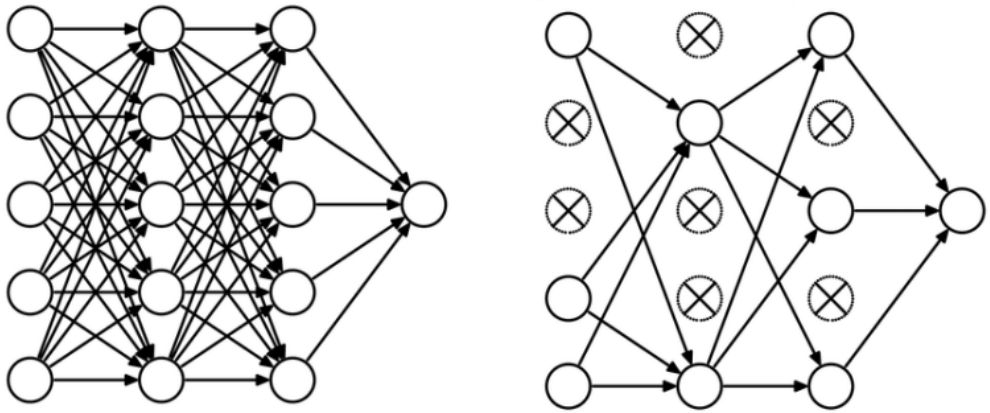
- Métodos *ensemble* são aqueles que combinam informações de diferentes preditores
  - Ao invés de treinar apenas um preditor, múltiplos preditores são treinados;
  - Supõe-se que cada preditor cometerá um tipo de erro e que vários preditores tenderão a errar menos, quando combinados;
  - Supõe-se que os erros de cada preditor sejam suprimidos pelos acertos dos outros;
  - Preditores devem produzir erros diferentes, de forma a cobrir todo o espaço possível
    - Essa condição evita que o erro de um preditor seja ratificado pelos demais [Goodfellow et al., 2016].

Dropout

- Dropout é um método de regularização que visa treinar todo o conjunto de sub-redes de uma rede neural;
  - Considera a rede como um conjunto de sub-redes que realizam ensemble;
- Para regularização, elimina alguns valores de camadas iniciais e intermediárias, por meio da multiplicação de pesos por zero;
  - Essa técnica visa treinar toda a rede, não apenas as partes que aprenderam no início;
  - Como a rede esquece algumas informações, é obrigada a aprender valores novos, reduzindo o *overfitting*;
  - Ao aprender informações novas, é capaz de melhor generalizar [Goodfellow et al., 2016].

Dropout

- A quantidade de neurônios que serão zerados pelo *dropout* é dada por um parâmetro definido antes do treinamento
  - Esse parâmetro indica o quanto a rede irá ignorar de seu aprendizado anterior para aprender novas informações [Srivastava et al., 2014]



Fonte: [Deshmukh, 2020]

Batch Normalization

- Batch normalization[14] é um método de regularização que visa manter cada camada da rede neural com média zero e variâncias unitárias, aumentando a convergência;
- O método consiste em diminuir as alterações das variáveis dependentes
  - Essa alteração impacta na distribuição das ativações da rede;
  - Para isso, as entradas de cada camada são normalizadas para cada *mini-batch* de treinamento;
- Permite, em diversos cenários, aumentar taxas de aprendizado, inicializar parâmetros com menos cuidado e remover outras técnicas de regularização [Ioffe and Szegedy, 2015].

[14] Tradução literal: normalização em lote.

Referências I

Breiman, L. (1996).
Bagging predictors.
Machine Learning, 24(2):123–140.

Chaudhary, M. (2020).
Activation functions: Sigmoid, tanh, relu, leaky relu, softmax.
[Online]; acessado em 26 de Janeiro de 2021. Disponível em: https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5.

Computer Science Wiki (2020).
Max-pooling / pooling.
[Online]; acessado em 08 de Setembro de 2020. Disponível em: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling.

Deshmukh, S. (2020).
Regularization in neural networks.
[Online]; acessado em 01 de Fevereiro de 2021. Disponível em: https://capablemachine.com/2020/08/20/regularization-in-neural-networks/.

Fawzi, A., Samulowitz, H., Turaga, D., and Frossard, P. (2016).
Adaptive data augmentation for image classification.
In 2016 IEEE International Conference on Image Processing (ICIP), pages 3688–3692.

Referências II

Florindo, J. a. B. (2018).
Redes neurais convolucionais.
[Online]; acessado em 08 de Setembro de 2020. Disponível em:
https://www.ime.unicamp.br/~jbflorindo/Teaching/2018/MT530/T10.pdf.

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep Learning.
MIT Press.
http://www.deeplearningbook.org.

Hansen, C. (2019).
Optimizers explained - adam, momentum and stochastic gradient descent.
[Online]; acessado em 27 de Janeiro de 2021. Disponível em:
https://mlfromscratch.com/optimizers-explained/#/.

Hoang Duong - Hoang Duong blog (2015).
Gradient descent and variants - convergence rate summary.
[Online]; acessado em 03 de Setembro de 2020. Disponível em:
http://hduongtrong.github.io/2015/11/23/coordinate-descent/.

IBM (2020).
Convolutional neural networks.
[Online]; acessado em 26 de Janeiro de 2021. Disponível em:
https://www.ibm.com/cloud/learn/convolutional-neural-networks.

Referências III

Ioffe, S. and Szegedy, C. (2015).
Batch normalization: Accelerating deep network training by reducing internal covariate shift.
In Bach, F. and Blei, D., editors, Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research, pages 448–456, Lille, France. PMLR.

Kopec, D. (2019).
Classic Computer Science Problems in Python.
Manning Publications Co, 1 edition.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).
Imagenet classification with deep convolutional neural networks.
In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12, pages 1097–1105, Red Hook, NY, USA. Curran Associates Inc.

Li, F.-F., Krishna, R., and Xu, D. (2021).
Convolutional neural networks (cnns / convnets).
[Online]; acessado em 26 de Janeiro de 2021. Disponível em:
`https://cs231n.github.io/convolutional-networks/`.

Marsland, S. (2014).
Machine Learning: An Algorithm Perspective.
CRC Press, 2 edition.
Disponível em: https://homepages.ecs.vuw.ac.nz/ marslast/MLbook.html.

Referências IV

Martin, S. (2018).
Whatâs the difference between a cnn and an rnn?
[Online]; acessado em 26 de Janeiro de 2021. Disponível em:
https://blogs.nvidia.com/blog/2018/09/05/whats-the-difference-between-a-cnn-and-an-rnn/.

Perez, L. and Wang, J. (2017).
The effectiveness of data augmentation in image classification using deep learning.
CoRR, abs/1712.04621.

Radford, A. (2014).
Visualizing optimization algos.
[Online]; acessado em 27 de Janeiro de 2021. Disponível em: https://imgur.com/a/Hqolp.

Rahman, R. (2017).
Visualising stochastic optimisers.
[Online]; acessado em 27 de Janeiro de 2021. Disponível em:
https://rnrahman.com/blog/visualising-stochastic-optimisers/.

Richert, W. and Coelho, L. P. (2013).
Building Machine Learning Systems with Python.
Packt Publishing Ltd., 1 edition.

Ruder, S. (2016).
An overview of gradient descent optimization algorithms.
CoRR, abs/1609.04747.

Referências V

Sadek Alaoui - SQLML (2017).
Convolutional neural network.
[Online]; acessado em 25 de Agosto de 2020. Disponível em:
http://sqlml.azurewebsites.net/2017/09/12/convolutional-neural-network/.

Shalev-Shwartz, S. and Ben-David, S. (2014).
Understanding Machine Learning: From Theory to Algorithms.
Cambridge University Press, 1 edition.
Disponível em: http://www.cs.huji.ac.il/ shais/UnderstandingMachineLearning.

Smith, S. W. (1997).
The Scientist and Engineer's Guide to Digital Signal Processing.
California Technical Publishing, San Diego, CA, USA.
http://www.dspguide.com.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014).
Dropout: A simple way to prevent neural networks from overfitting.
15:1929–1958.

Upadhyay, Y. (2019).
Regularization techniques for neural networks.
[Online]; acessado em 01 de Fevereiro de 2021. Disponível em: https://towardsdatascience.com/regularization-techniques-for-neural-networks-e55f295f2866.

Weisstein, E. W. (2018).
Convolution.
http://mathworld.wolfram.com/Convolution.html.

Referências VI


Wikipedia contributors (2020a).
Convolution.
[Online]; acessado em 08 de Setembro de 2020. Disponível em: https://en.wikipedia.org/wiki/Convolution.


Wikipedia contributors (2020b).
Rectifier (neural networks).
[Online]; acessado em 08 de Setembro de 2020. Disponível em: https://en.wikipedia.org/wiki/Rectifier_(neural_networks).


Wikipedia contributors (Charles J Sharp) (2020).
Panthera onca.
[Online]; acessado em 09 de Setembro de 2020. Disponível em: https://pt.wikipedia.org/wiki/Panthera_onca.


Wikipedia contributors (Dominique Jacholke) (2020).
Gato.
[Online]; acessado em 09 de Setembro de 2020. Disponível em: https://pt.wikipedia.org/wiki/Gato.


Wikipedia contributors (Harrie Gielen) (2020).
Leão.
[Online]; acessado em 09 de Setembro de 2020. Disponível em: https://pt.wikipedia.org/wiki/Le%C3%A3o.


Wikipedia contributors (Tony Hisgett) (2020).
Tigre.
[Online]; acessado em 09 de Setembro de 2020. Disponível em: https://pt.wikipedia.org/wiki/Tigre.

Referências VII

Wong, S. C., Gatt, A., Stamatescu, V., and McDonnell, M. D. (2016).
Understanding data augmentation for classification: when to warp?
2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), abs/1609.08764.

Yakout, A. (2021).
Coursera deep learning specialization by andrew ng.
[Online]; acessado em 27 de Janeiro de 2021. Disponível em: https://yakout.io/deeplearning/coursera-deep-learning-course-2-week-2/.

Yunus, M. (2020).
11 artificial neural network (ann) â part 6 konsep dasar convolutional neural network (cnn).
[Online]; acessado em 26 de Janeiro de 2021. Disponível em: https://yunusmuhammad007.medium.com/11-artificial-neural-network-ann-part-6-konsep-dasar-convolutional-neural-network-cnn-3cc10fd9cf6