

Grupo de Estudos de Visão Computacional

Nomes dos apresentadores

Daniel Machado

Gustavo Novack

Pedro Martins

Título

Pré-Processamento de Dados,
Regularização e Otimização de
Redes Neurais

Goiânia

2024

INF

INSTITUTO DE
INFORMÁTICA

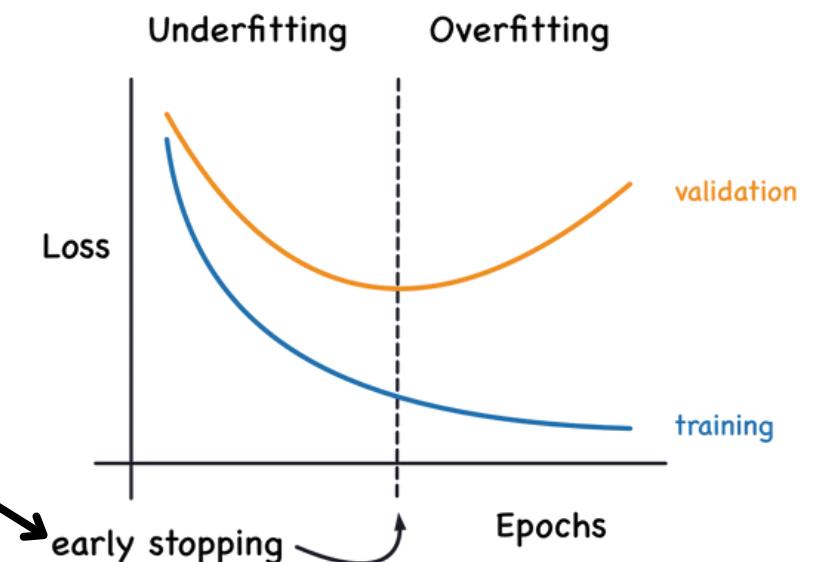
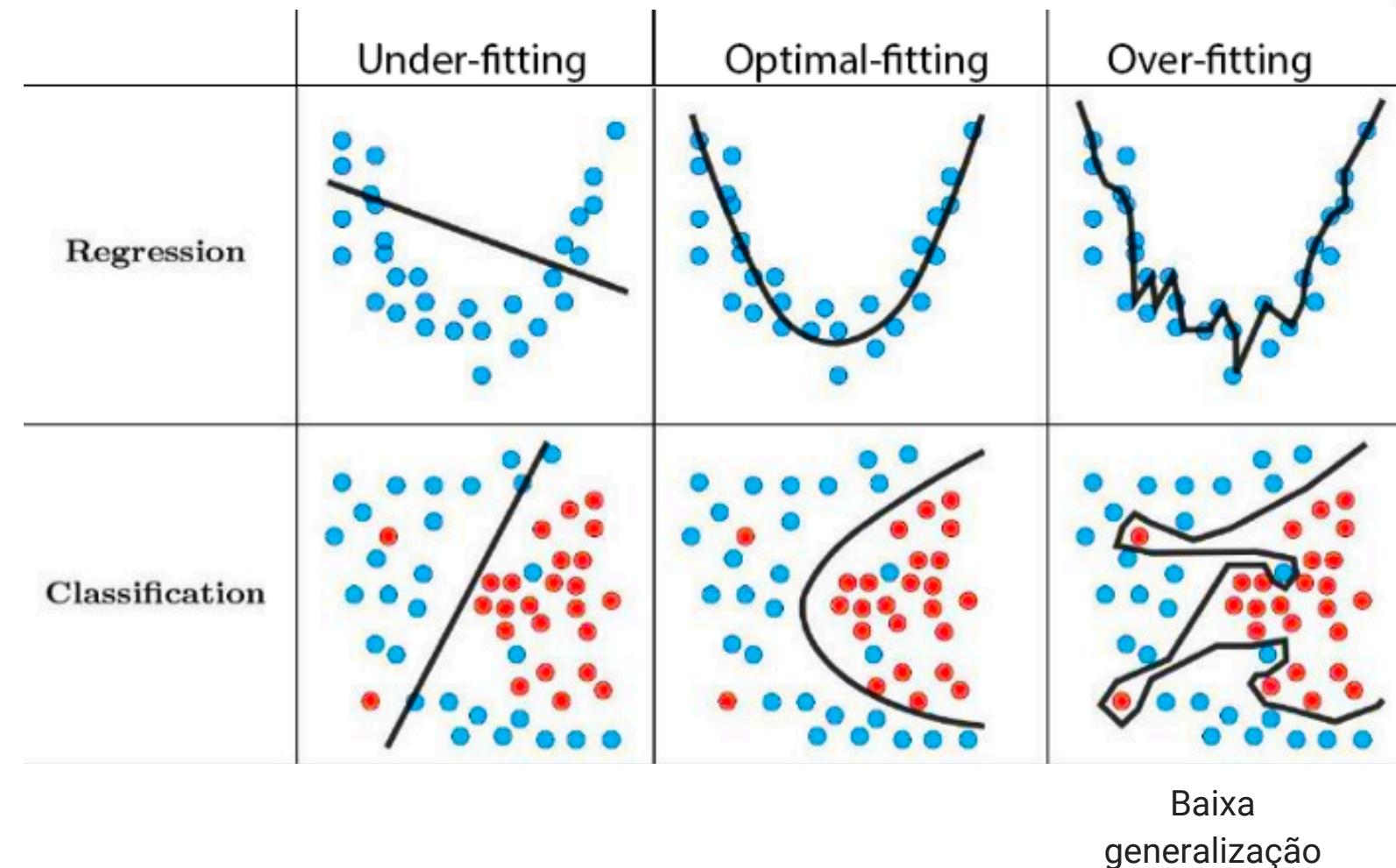
Apoio



Regularização e Otimização em Redes Neurais para CV

Recapitulando ...

1. Por que prevenir overfitting é essencial em visão computacional?
2. Conexão entre regularização, otimizadores e o desempenho de redes.



Regularização e Otimização em Redes Neurais para CV

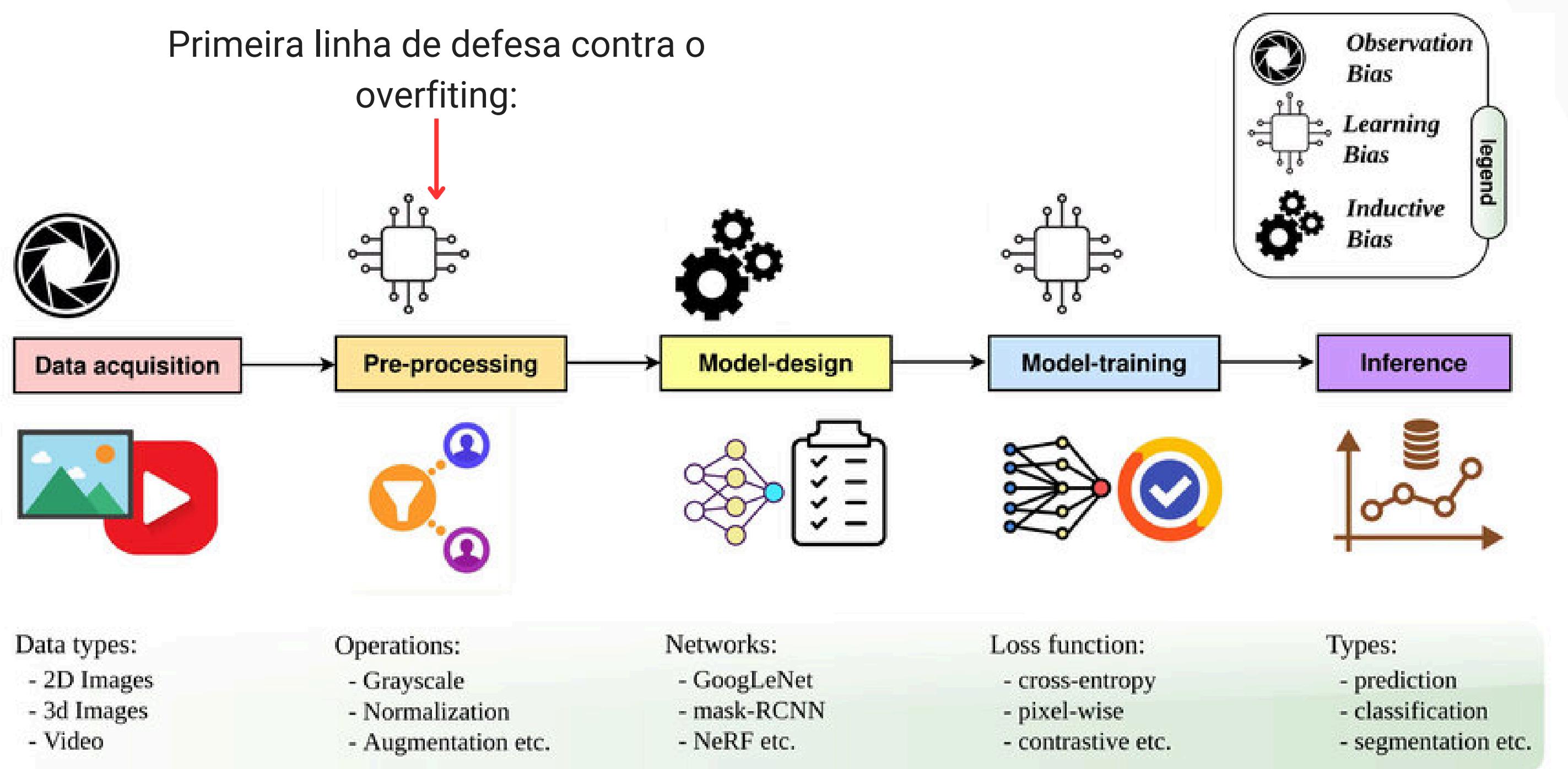
Senhores, teremos
300 Hiperparâmetros...



contra 30000 overfittings!

Regularização e Otimização em Redes Neurais para CV

Pipeline de Visão Computacional:



Regularização e Otimização em Redes Neurais para CV

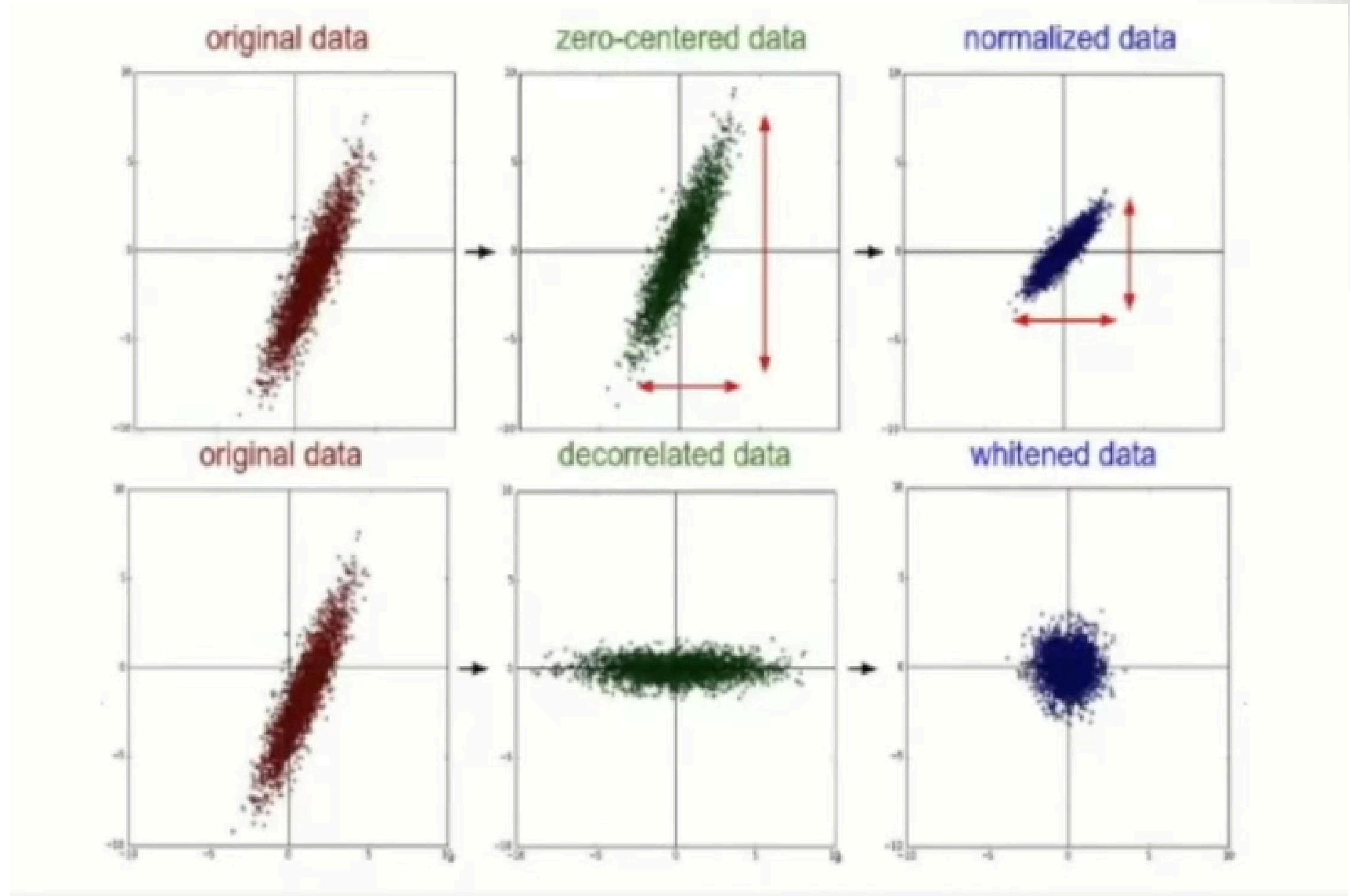
Dados mal processados têm escalas e distribuições inconsistentes, o que pode afetar:

1. Convergência durante o treinamento: Dados com escalas muito diferentes levam a gradientes desbalanceados, dificultando a atualização uniforme dos pesos da rede.
2. Generalização: Modelos treinados com dados mal escalados podem aprender padrões errados (overfitting em valores extremos ou insignificância de dimensões importantes).

Exemplo comum: Imagens em escalas RGB com valores variando entre 0-255 (intensidades de pixel). Essas escalas altas podem produzir gradientes enormes e dificultar a convergência.

Regularização e Otimização em Redes Neurais para CV

Alguns tipos de processamento:



Regularização e Otimização em Redes Neurais para CV

Algumas das técnicas mais comuns de Pré-processamento:

Subtrai-se a média (μ) dos valores dos pixels e divide-se pelo desvio padrão (σ).

$$X_{std} = \frac{X - \mu}{\sigma}$$

Normalização:

Gradiente em relação a 'w'

$$\nabla_w L = X^T \cdot (h(X) - Y)$$

Evitar desbalanço nos pesos:

Sendo X o tensor de entrada, caso X tenha alguma dimensão com valores altos (ex.: $x_1 \sim 10^3$) e outra com valores baixos (ex.: $x_2 \sim 10^{-1}$) a atualização dos pesos será dominada pela dimensão x_1 , causando insignificância na x_2

Regularização e Otimização em Redes Neurais para CV

Normalização:

- Sem normalização:

$$X^T X = \begin{bmatrix} 140000 & 14 \\ 14 & 0.0014 \end{bmatrix}$$

- Altamente desbalanceada. Gradiente será dominado por x_1 .
- Com normalização:

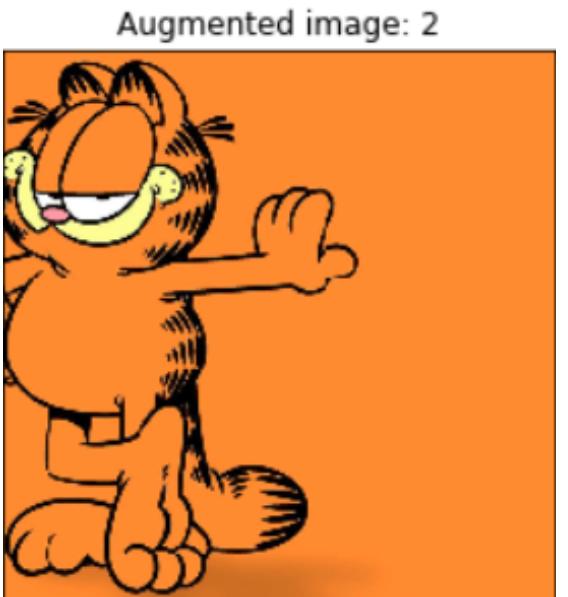
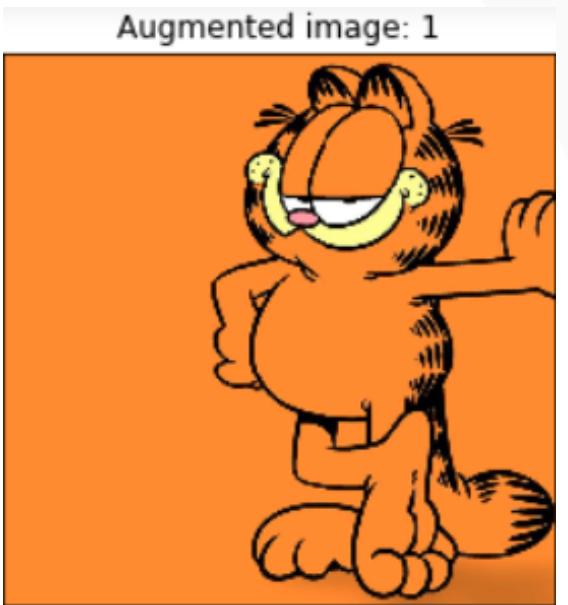
$$X' = \frac{X - \mu}{\sigma} \quad \rightarrow \quad X'^T X' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Balanceada. Gradientes têm magnitudes semelhantes.

Regularização e Otimização em Redes Neurais para CV

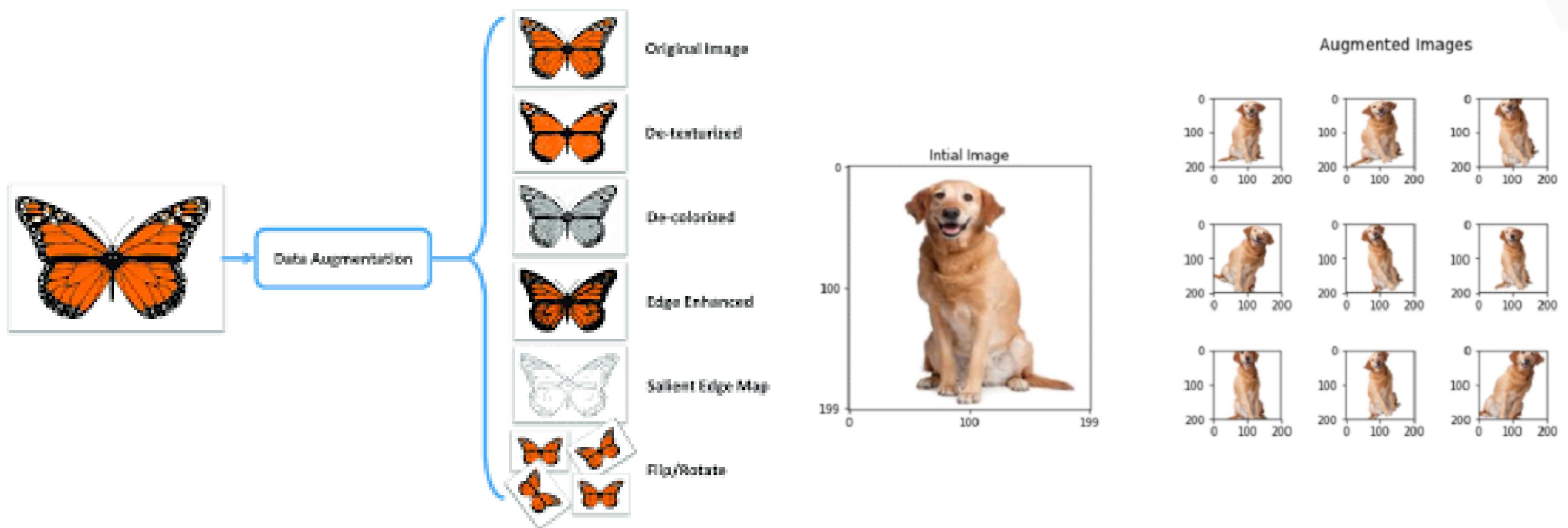
Data Augmentation:

- 1. Aumentar o dataset:** Quando os dados disponíveis são limitados, gera-se mais variações sem necessidade de coleta ou rotulação adicional.
- 2. Prevenir overfitting:** Os modelos aprendem padrões mais generalizáveis ao invés de memorizar o dataset.
- 3. Melhorar a robustez do modelo:** O modelo se torna mais tolerante a variações no mundo real (ruídos, iluminações diferentes, etc.).
- 4. Simular cenários do mundo real:** Muitos problemas têm variabilidade nas condições de captura (como câmeras diferentes ou ângulos variados).



Regularização e Otimização em Redes Neurais para CV

Data Augmentation:



Regularização e Otimização em Redes Neurais para CV

Regularização: O problema do overfitting

Regularização funciona limitando a capacidade da rede de se ajustar perfeitamente ao conjunto de treino, incentivando o modelo a priorizar padrões mais gerais e relevantes. Isso é feito introduzindo penalizações ou restrições durante o treinamento.

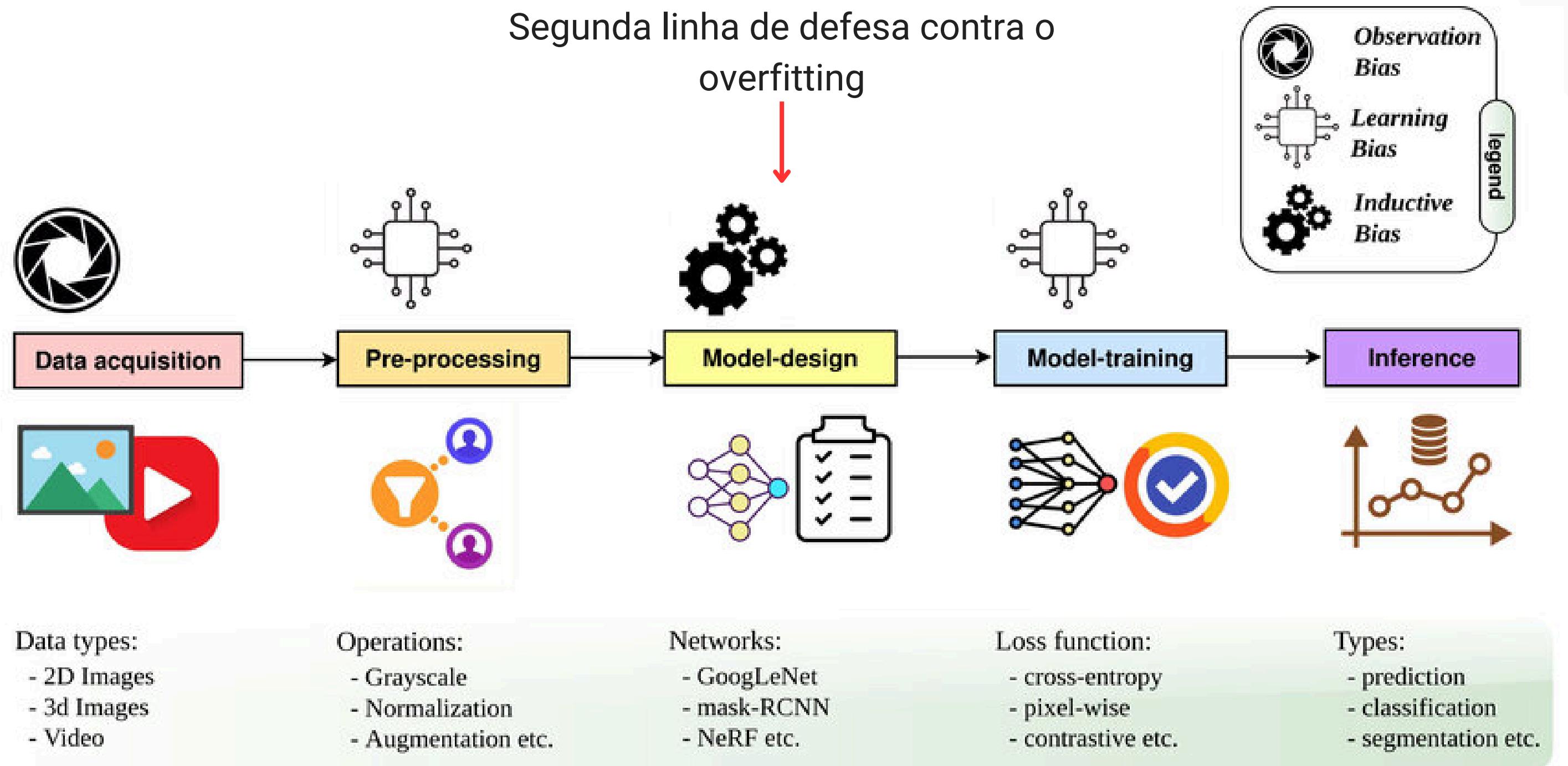
Quando e Onde a Regularização Atua?

Regularização é aplicada na fase de treinamento de uma rede neural. Ela atua diretamente nos pesos da rede, modificando a função de perda ou o comportamento do modelo durante o aprendizado. As operações de regularização podem ser implementadas nas seguintes etapas:

- **Função de perda:** Incorporando termos de penalização (e.g., L1, L2).
- **Atualização dos pesos:** Controlando como os parâmetros da rede são ajustados (e.g., Dropout).
- **Treinamento e preprocessamento:** Introduzindo técnicas como data augmentation.

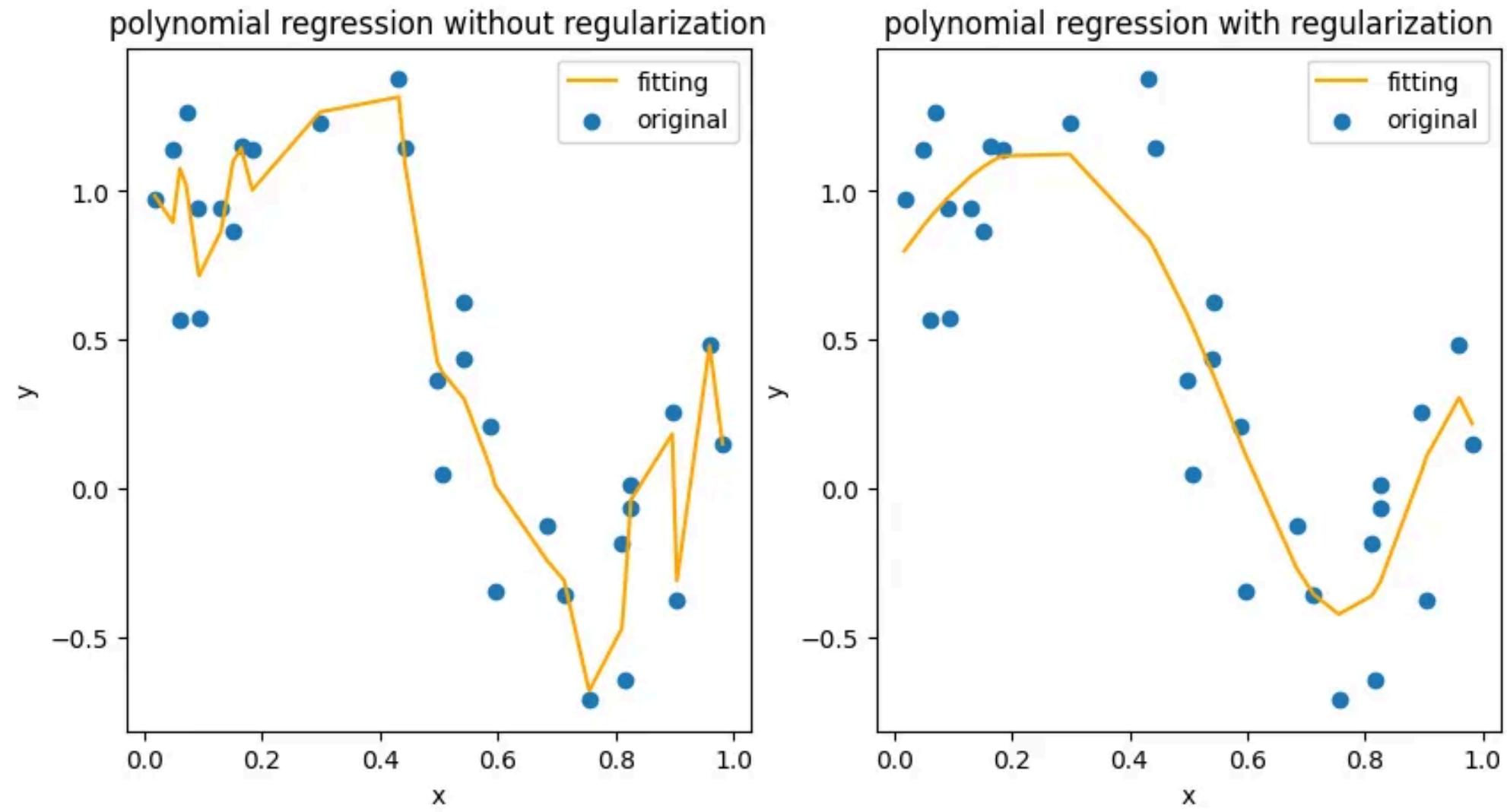
Regularização e Otimização em Redes Neurais para CV

Regularização: O problema do overfitting



Regularização e Otimização em Redes Neurais para CV

Regularização: O problema do overfitting



Polynomial regression:

$$\begin{aligned}y &= \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_{15} x^{15} \\&= \sum_{p=0}^{15} \beta_p x^p\end{aligned}$$

Regularização e Otimização em Redes Neurais para CV

Regularização: O problema do overfitting

Adicionamos o termo de regularização à função objetivo e otimizamos os parâmetros com base nele.

General regularization formulation

$$\mathcal{L}(\mathbf{w}) + \lambda R(\mathbf{w})$$

λ : Regularization strength

\mathbf{w} : the model parameters or weights

$\mathcal{L}(\mathbf{w})$: Objective function or loss function

$R(\mathbf{w})$: Regularization term or regularizer or penalty

Ex) In the previous example case, we can describe the above formula as:

$$\mathcal{L}(\boldsymbol{\beta}) + \lambda R(\boldsymbol{\beta}) = \boxed{\frac{1}{n} \sum_{i=1}^n \left(y_i - \left(\sum_{p=0}^{15} \beta_p x_i^p \right) \right)^2} + \boxed{\lambda \sum_{p=0}^{15} \|\beta_p\|^2}$$

Least mean square error
※ the error between the original and predicted data

Regularization

Regularização e Otimização em Redes Neurais para CV

Regularização:O problema do overfitting

```
# polynomial regression without regularization
poly_regression_model.coef_
```

```
array([ 0.0000000e+00, -1.22941033e+03,  5.24297034e+04, -1.14665464e+06,
       1.48641044e+07, -1.23988729e+08,  7.01323988e+08, -2.78469946e+09,
       7.93342013e+09, -1.64004162e+10,  2.46268195e+10, -2.65787326e+10,
      2.00820965e+10, -1.00801528e+10,  3.01856498e+09, -4.08003987e+08])
```

```
# polynomial regression with regularization
ridge_model.coef_
```

```
array([ 0.          ,  2.87867528, -3.59044009, -5.74529316, -2.18761267,
       1.40920191,  3.31384599,  3.68429763,  3.10603754,  2.09071807,
       0.97688102, -0.04580222, -0.88969133, -1.52931975, -1.97362039,
      -2.24833423])
```

Regularização e Otimização em Redes Neurais para CV

1. Regularização L1 e L2 (Penalização dos Pesos)

- **Como funciona:** Adiciona um termo de penalização aos pesos na função de perda.
- **Operações Matemáticas:**
 - **L1 (Regularização Lasso):**

$$\text{Função de perda regularizada} = \mathcal{L}(y, \hat{y}) + \lambda \sum_i |w_i|$$

Incentiva pesos pequenos ou nulos, promovendo **sparsity** (esparsidade).

- **L2 (Regularização Ridge):**

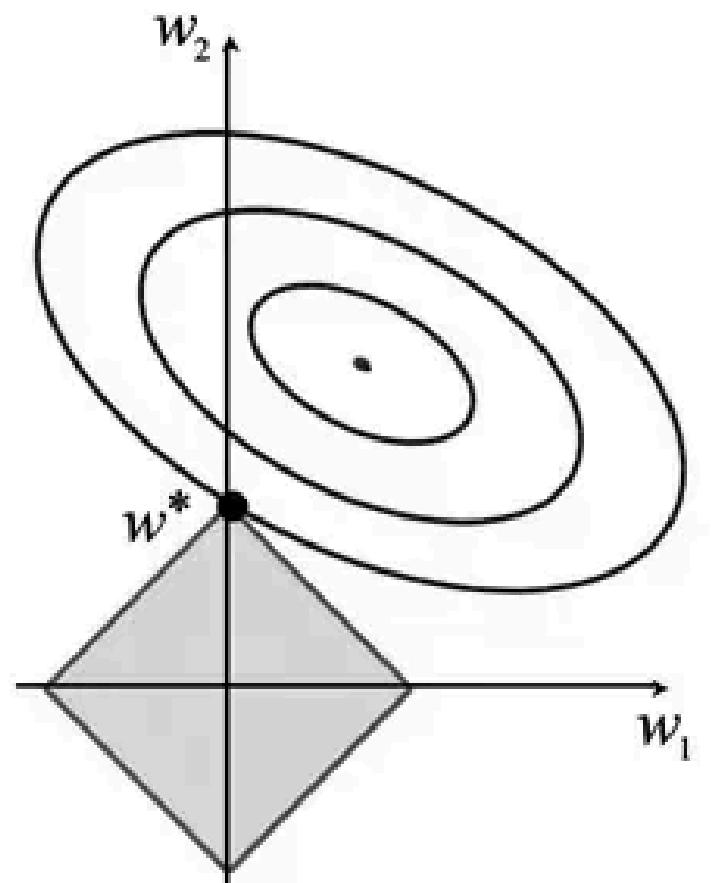
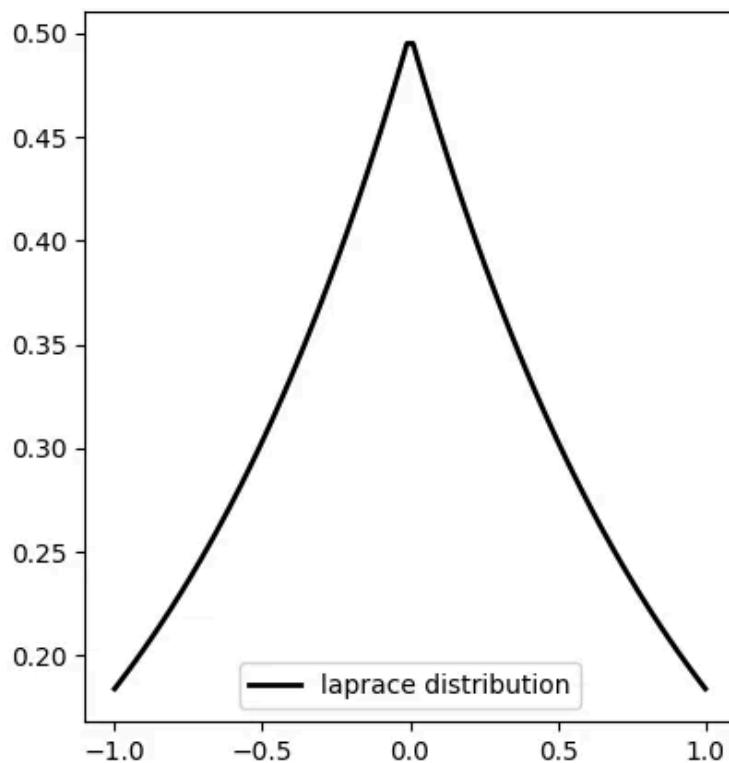
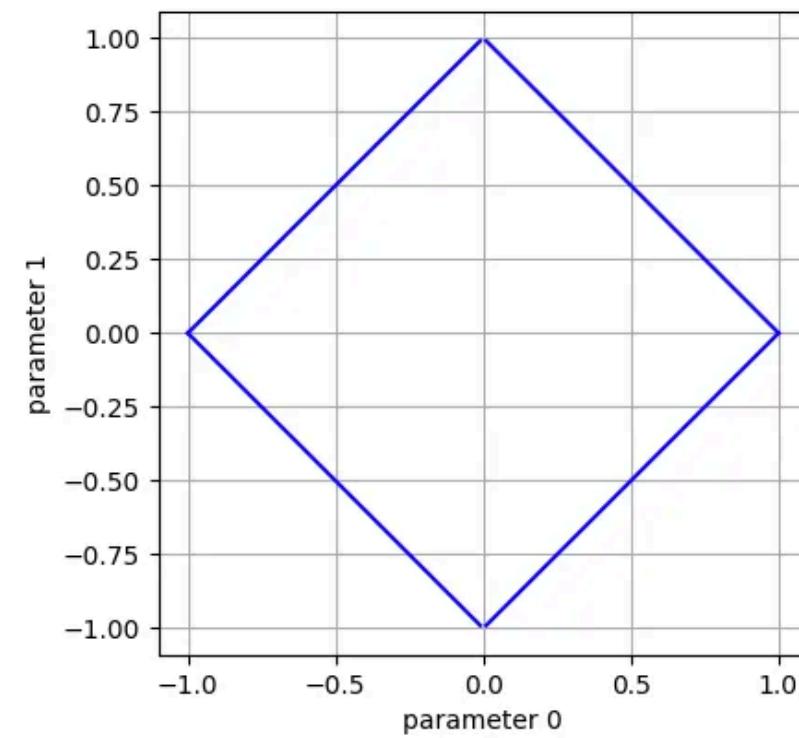
$$\text{Função de perda regularizada} = \mathcal{L}(y, \hat{y}) + \lambda \sum_i w_i^2$$

Penaliza pesos grandes, incentivando uma distribuição uniforme dos parâmetros.

- **Efeitos:** Evita ajustes excessivos ao conjunto de treino.

Regularização e Otimização em Redes Neurais para CV

Regularização Lasso (L1)



L1

Regularização e Otimização em Redes Neurais para CV

Regularização Ridge (L2)

- **Definição Matemática:** A regularização L2 adiciona a soma dos quadrados dos pesos:

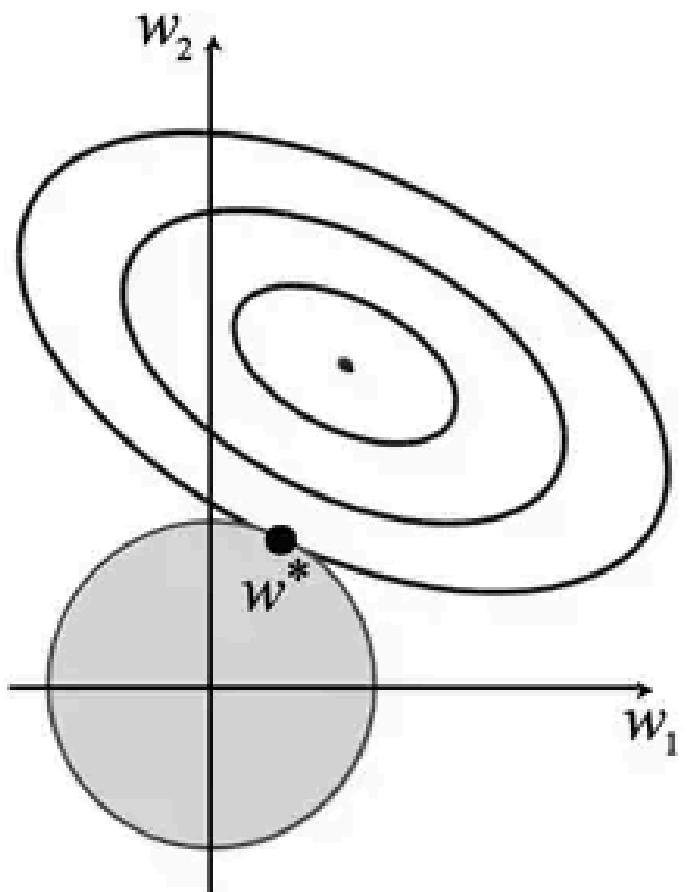
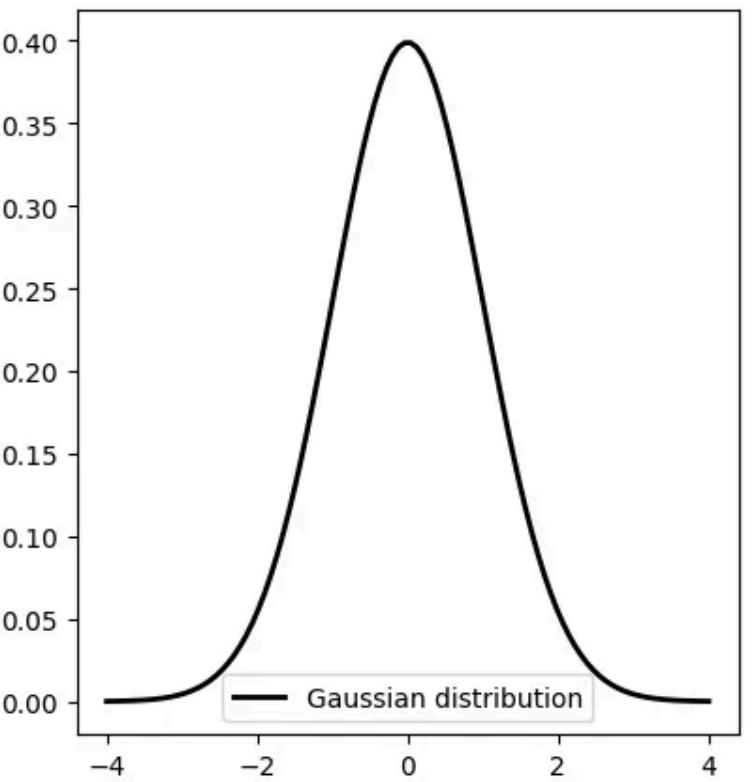
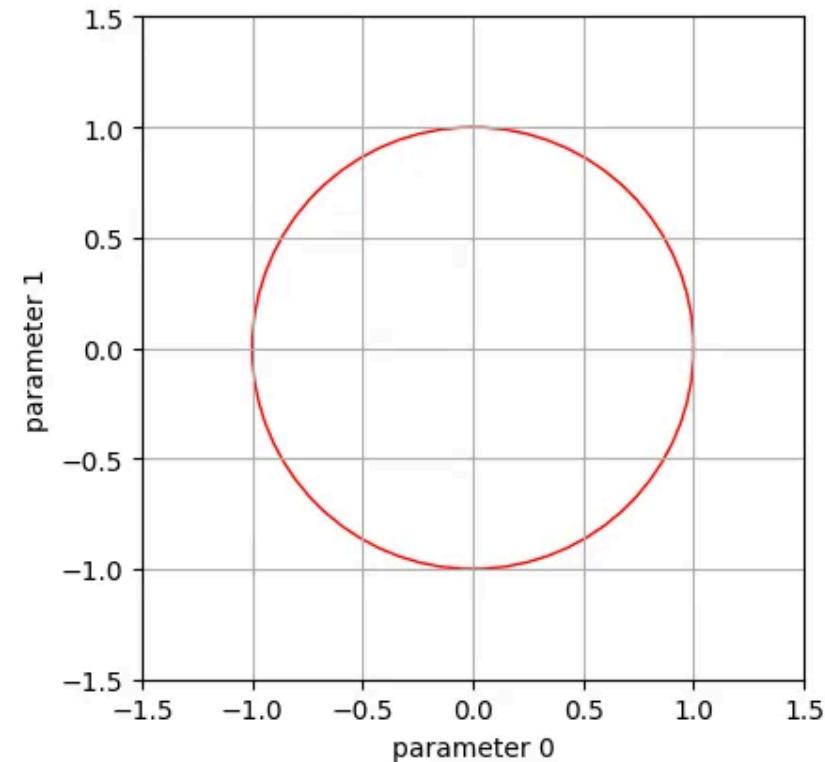
$$\mathcal{L}_{\text{total}} = \mathcal{L}(y, \hat{y}) + \lambda \sum_{i,j} W_{i,j}^2$$

- **Efeito Matemático:** A penalização quadrática reduz o valor absoluto dos pesos grandes, mas sem forçá-los a zero. Isso encoraja uma distribuição mais uniforme dos pesos nos filtros convolucionais.

Agora, a Loss penaliza coeficientes muito altos!!!

Regularização e Otimização em Redes Neurais para CV

Regularização Ridge (L2)



L2

Regularização e Otimização em Redes Neurais para CV

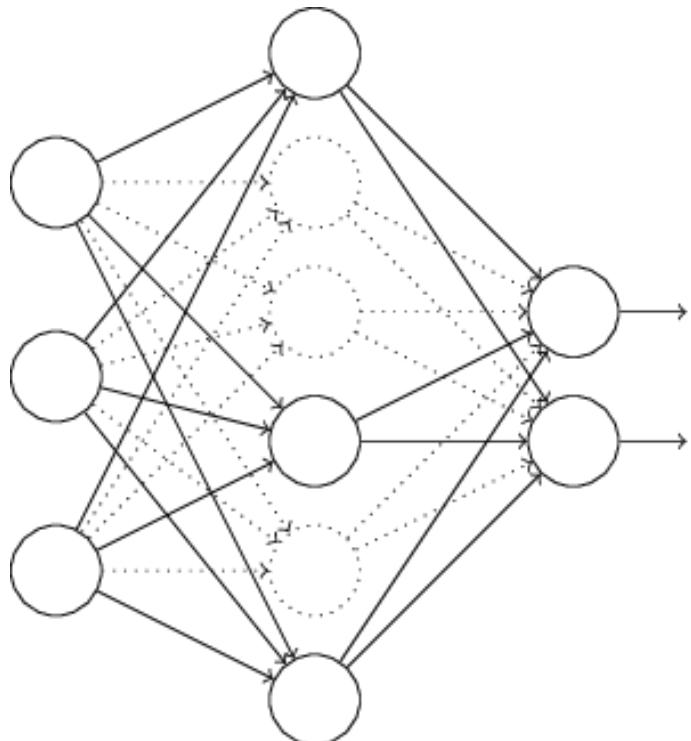
Comparação: Quando Usar L1 ou L2?

Característica	L1	L2
Impacto nos Pesos	Esparsidade	Suavização
Dataset com Padrões Redundantes	Excelente	Bom
Dados de Alta Dimensionalidade	Reduz dimensões	Mantém todas as conexões
Exemplo de Uso	Compressão de modelos	Classificação geral

Regularização e Otimização em Redes Neurais para CV

Regularização: Dropout

O Dropout funciona **desativando aleatoriamente** neurônios durante o treinamento, forçando a rede a aprender representações mais robustas e menos dependentes de neurônios específicos.

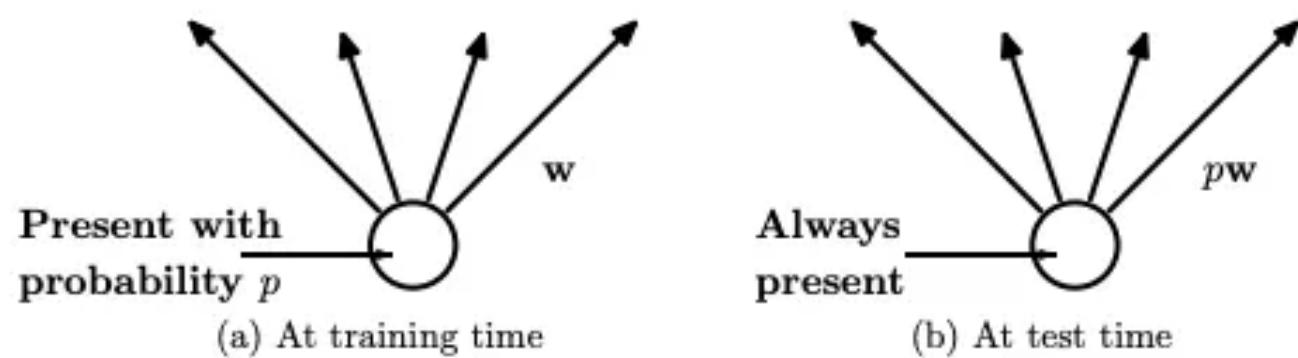


Regularização e Otimização em Redes Neurais para CV

Durante cada iteração do treino, o neurônio estará ativo ou não com uma probabilidade p .

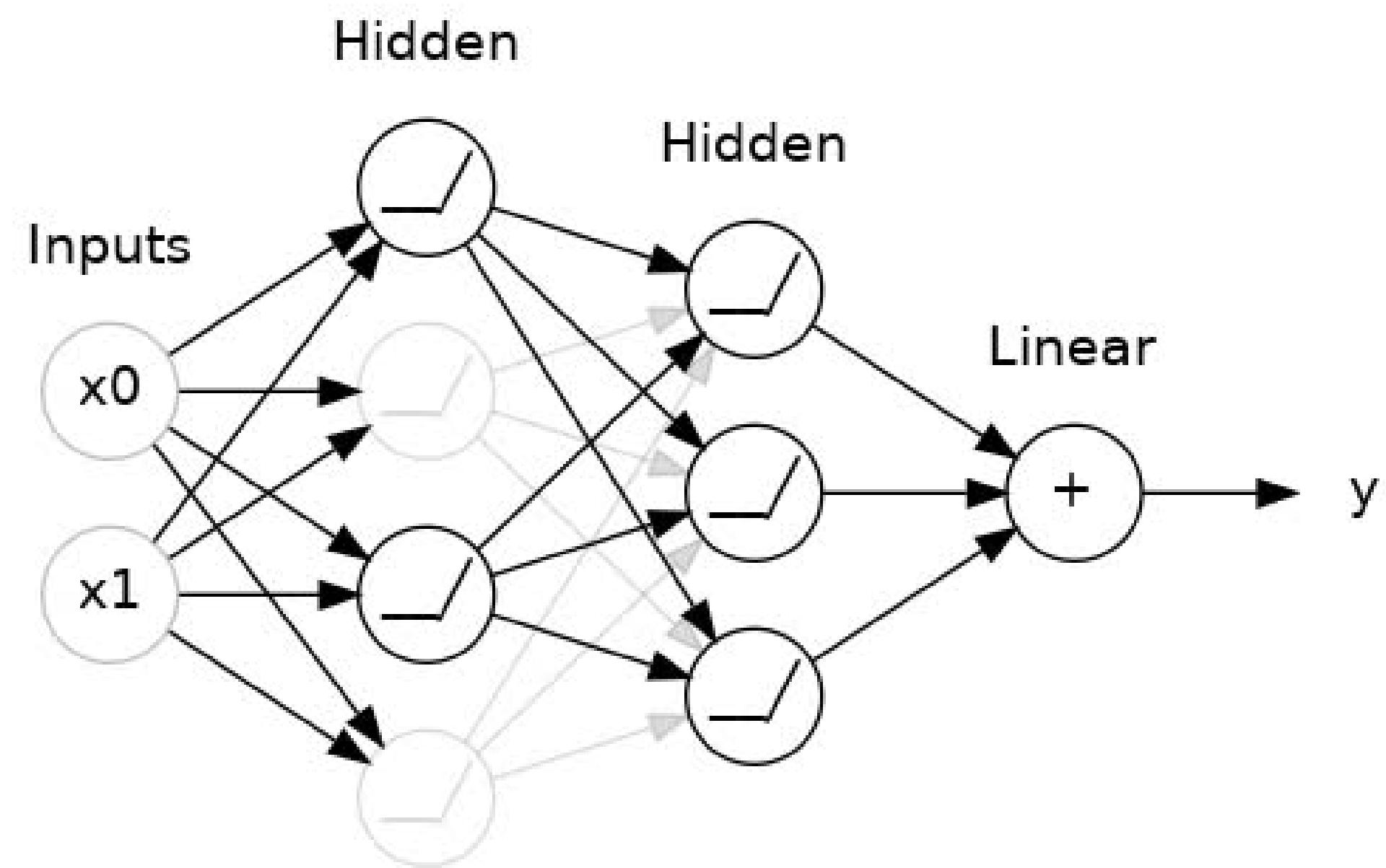
$$h' = \begin{cases} 0 & \text{with probability } p \\ \frac{h}{1-p} & \text{otherwise} \end{cases}$$

Mas na inferência, ele sempre estará presente.



Regularização e Otimização em Redes Neurais para CV

Exemplo visual:



Otimização de redes neurais:

1.1. O que é Otimização em Redes Neurais?

O objetivo da otimização em redes neurais é encontrar os valores dos pesos w e vieses b que minimizam uma função de perda $\mathcal{L}(w, b)$. Essa função mede o erro entre as previsões do modelo e os valores reais.

Regularização e Otimização em Redes Neurais para CV

No último episódio...

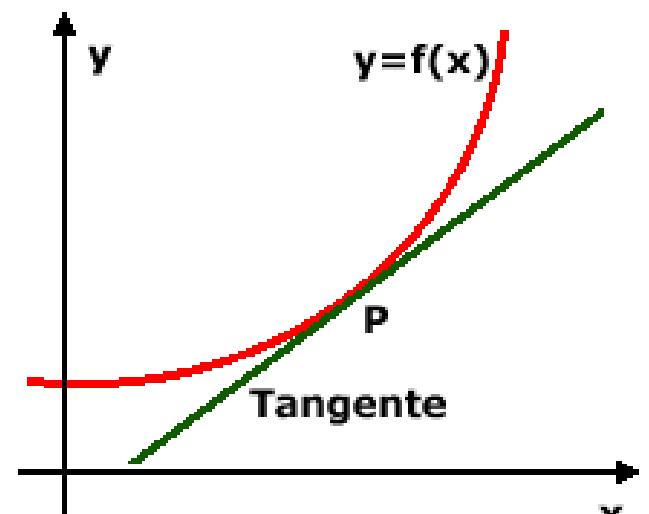
1.2. Gradiente e Gradiente Descendente

- **Gradiente:** O gradiente $\nabla \mathcal{L}(\mathbf{w})$ é o vetor de derivadas parciais da função de perda em relação a cada peso. Ele aponta na direção do maior aumento da função de perda.
- **Gradiente Descendente:** O gradiente descendente atualiza os pesos movendo-os na direção oposta ao gradiente para reduzir a perda:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \mathcal{L}(\mathbf{w})$$

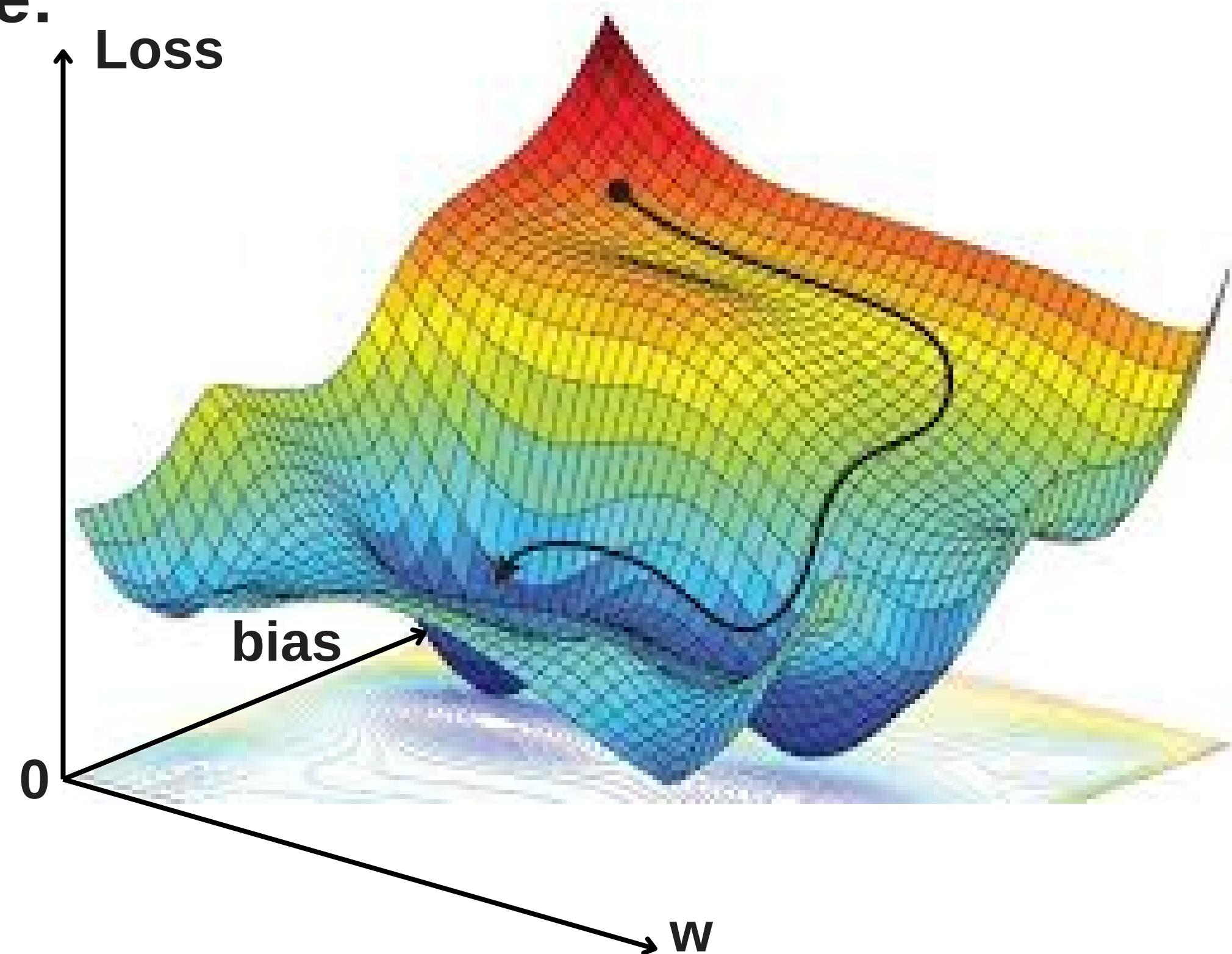
Onde:

- η : Taxa de aprendizado, que controla o tamanho dos passos.



Regularização e Otimização em Redes Neurais para CV

Visualmente:



Regularização e Otimização em Redes Neurais para CV

2. Principais Otimizadores

2.1. Stochastic Gradient Descent (SGD)

2.1.1. Visão Geral

O SGD é uma variante do gradiente descendente que calcula o gradiente e atualiza os pesos usando apenas um minibatch de dados por vez, em vez de usar o conjunto inteiro.

2.1.2. Fórmula

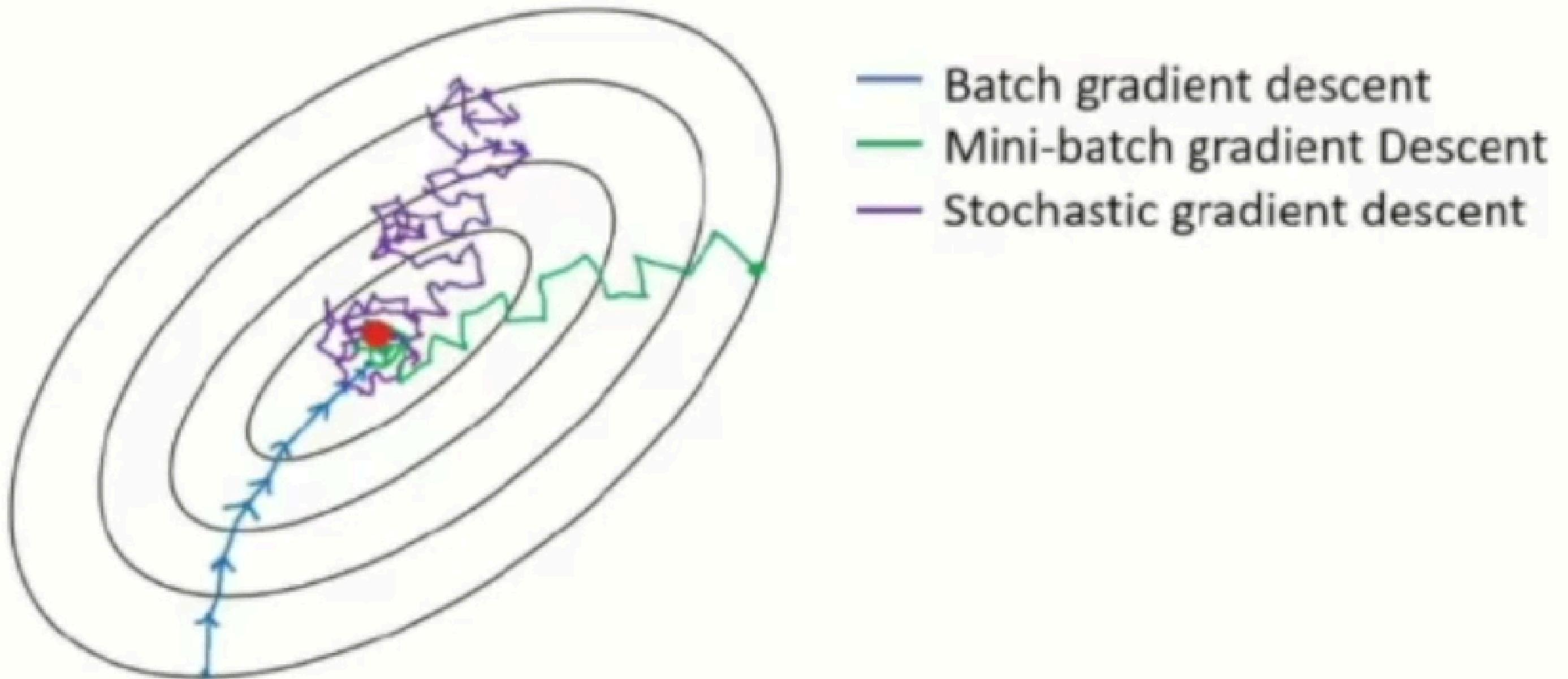
Para cada minibatch \mathcal{B} , o SGD atualiza os pesos como:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \mathcal{L}_{\mathcal{B}}(\mathbf{w})$$

Onde $\nabla \mathcal{L}_{\mathcal{B}}(\mathbf{w})$ é o gradiente calculado no minibatch.

Regularização e Otimização em Redes Neurais para CV

Visualmente:



Regularização e Otimização em Redes Neurais para CV

2.1.3. Vantagens e Desvantagens

- **Vantagens:**

- Mais rápido que o gradiente descendente completo em grandes conjuntos de dados.
- Permite escapar de mínimos locais rasos devido à sua natureza estocástica.

- **Desvantagens:**

- Convergência instável devido a flutuações no gradiente.
- Sensível à escolha da taxa de aprendizado.

Regularização e Otimização em Redes Neurais para CV

Adam:

Visão Geral sobre Adam

O Adam (*Adaptive Moment Estimation*) é um dos otimizadores mais populares em redes neurais devido à sua eficiência e capacidade de adaptação. Ele combina as ideias de dois métodos anteriores:

1. Momentum: Utiliza o histórico dos gradientes para suavizar o processo de otimização, ajudando a acelerar a convergência e reduzir oscilações.
2. RMSProp: Ajusta a taxa de aprendizado individualmente para cada parâmetro com base em uma média móvel dos gradientes ao longo do tempo.

A principal inovação do Adam é combinar esses dois conceitos, estimando dinamicamente os momentos de primeira ordem (média dos gradientes) e segunda ordem (variância dos gradientes). Isso permite que Adam adapte a taxa de aprendizado para cada parâmetro durante o treinamento, tornando-o especialmente eficaz em problemas com gradientes complexos e em datasets ruidosos.

Regularização e Otimização em Redes Neurais para CV

Adam:

Intuição Simples

Imagine que o Adam é como um motorista que ajusta a velocidade do carro com base em duas coisas:

1. Direção atual (primeira ordem): Ele observa para onde o carro está indo.
2. Obstáculos e velocidade média (segunda ordem): Ele considera o quanto difícil está a estrada, ajustando o quanto acelerar ou frear.

O Adam faz isso de forma inteligente para cada peso da rede neural, ajudando a alcançar a solução de forma eficiente e estável.

Regularização e Otimização em Redes Neurais para CV

Passo a Passo do Adam

1. **Gradiente Atual:** Após calcular o gradiente da função de perda em relação aos pesos, o Adam utiliza dois tipos de estimativas:
 - **Primeira Ordem (m_t):** Representa a direção geral do gradiente ao longo do tempo (média móvel dos gradientes).
 - **Segunda Ordem (v_t):** Representa a "energia" ou variabilidade do gradiente (média móvel dos quadrados dos gradientes).
2. **Correção de Viés:** Como m_t e v_t começam pequenos no início do treinamento, Adam aplica uma correção para compensar o viés inicial, garantindo que as atualizações sejam precisas.
3. **Atualização dos Pesos:** Cada peso é atualizado considerando essas duas estimativas, com uma taxa de aprendizado ajustada dinamicamente.

Regularização e Otimização em Redes Neurais para CV

2.2.2. Fórmulas

Adam atualiza os pesos usando as seguintes equações:

1. Cálculo dos momentos:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \nabla \mathcal{L}(\mathbf{w}_t)$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) (\nabla \mathcal{L}(\mathbf{w}_t))^2$$

Onde:

- \mathbf{m}_t : Momento de primeira ordem (média dos gradientes).
- \mathbf{v}_t : Momento de segunda ordem (média dos quadrados dos gradientes).
- β_1, β_2 : Taxas de decaimento exponencial ($\beta_1 = 0.9, \beta_2 = 0.999$).

2. Correção de viés:

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}$$

3. Atualização dos pesos:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \hat{\mathbf{m}}_t$$

Onde ϵ é um pequeno valor para evitar divisões por zero (geralmente 10^{-8}).

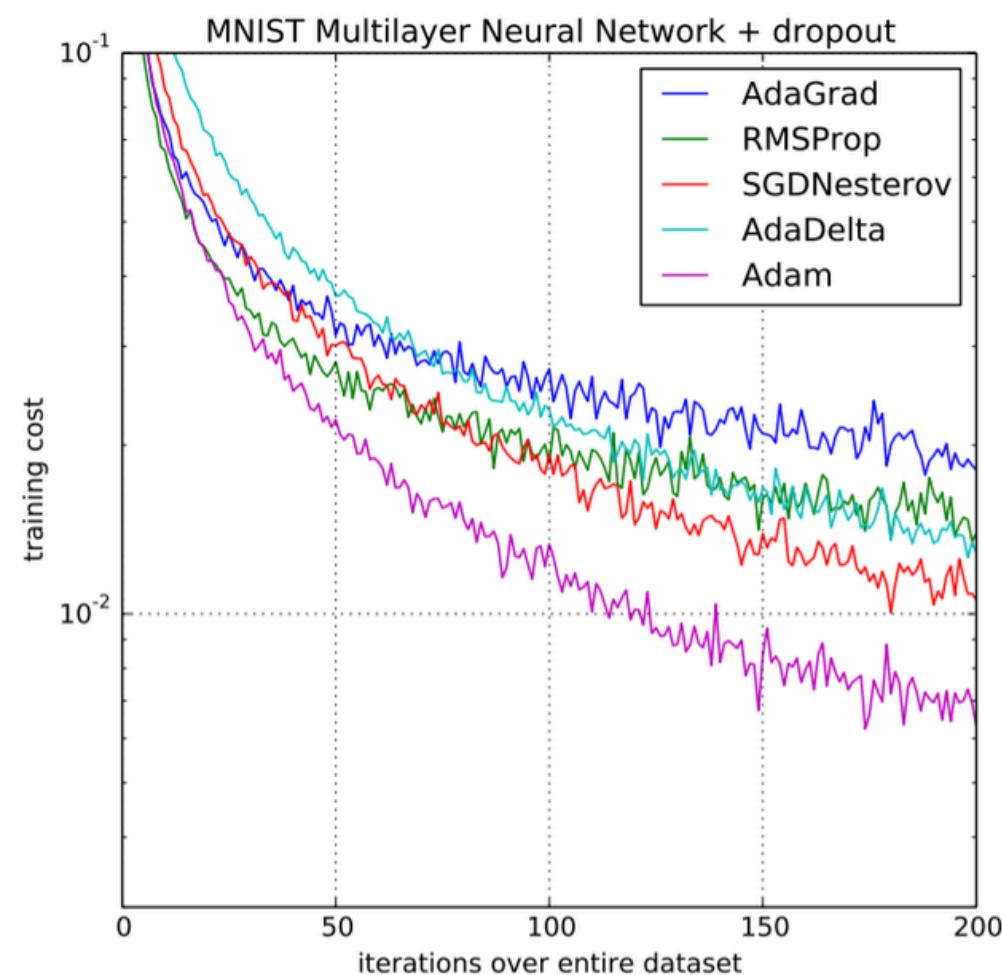
Regularização e Otimização em Redes Neurais para CV

Por Que o Adam Funciona Bem?

1. **Adaptação Automática:** Ele ajusta a taxa de aprendizado para cada peso, acelerando nas dimensões que estão aprendendo lentamente e desacelerando nas que estão mudando rápido.
2. **Estabilidade em Dados Ruidosos:** As estimativas suavizadas dos gradientes ajudam a evitar flutuações excessivas, comuns em métodos como SGD puro.
3. **Convergência Rápida:** Sua abordagem adaptativa tende a encontrar bons mínimos rapidamente, mesmo em problemas não estacionários (com gradientes que mudam ao longo do tempo).

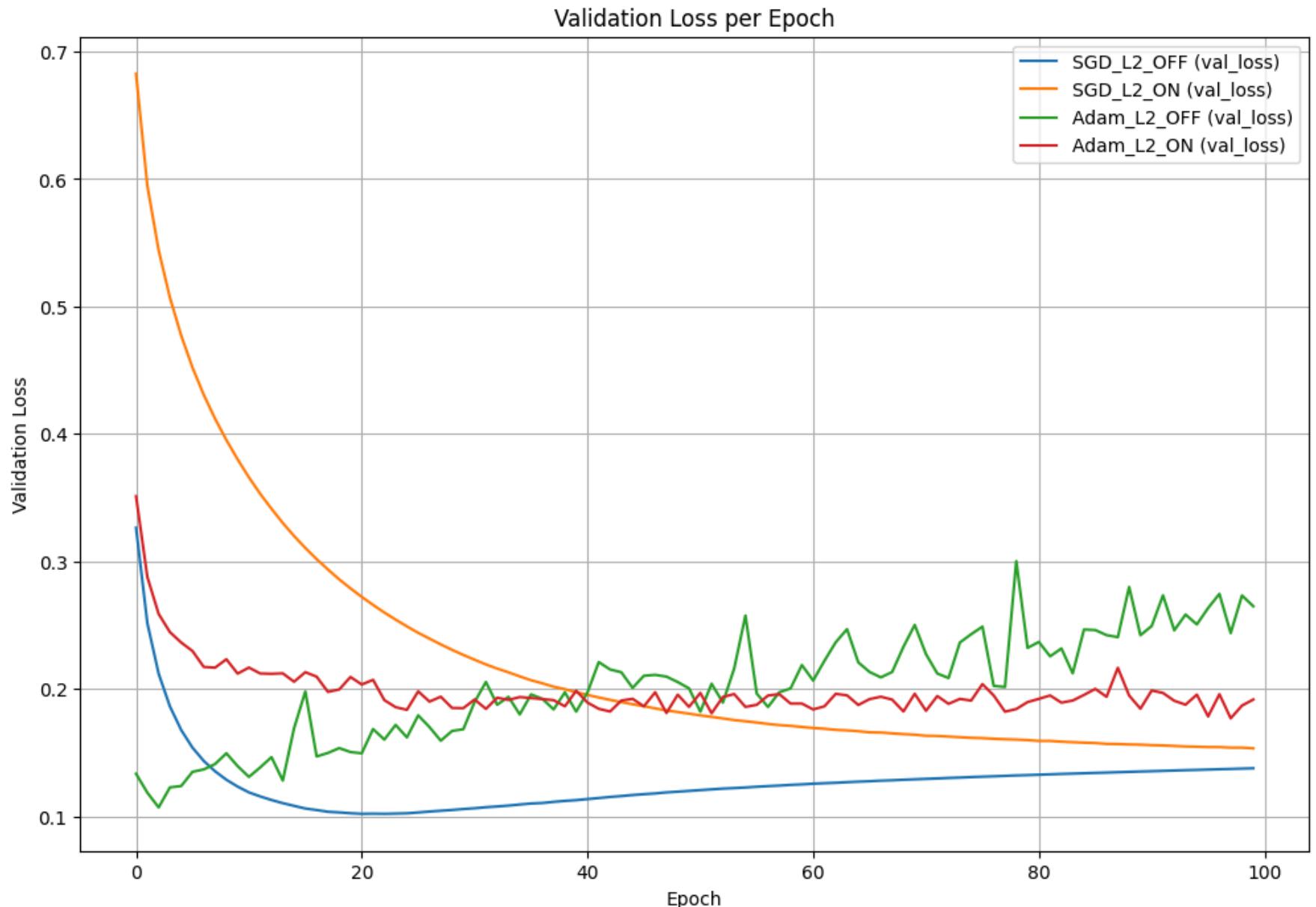
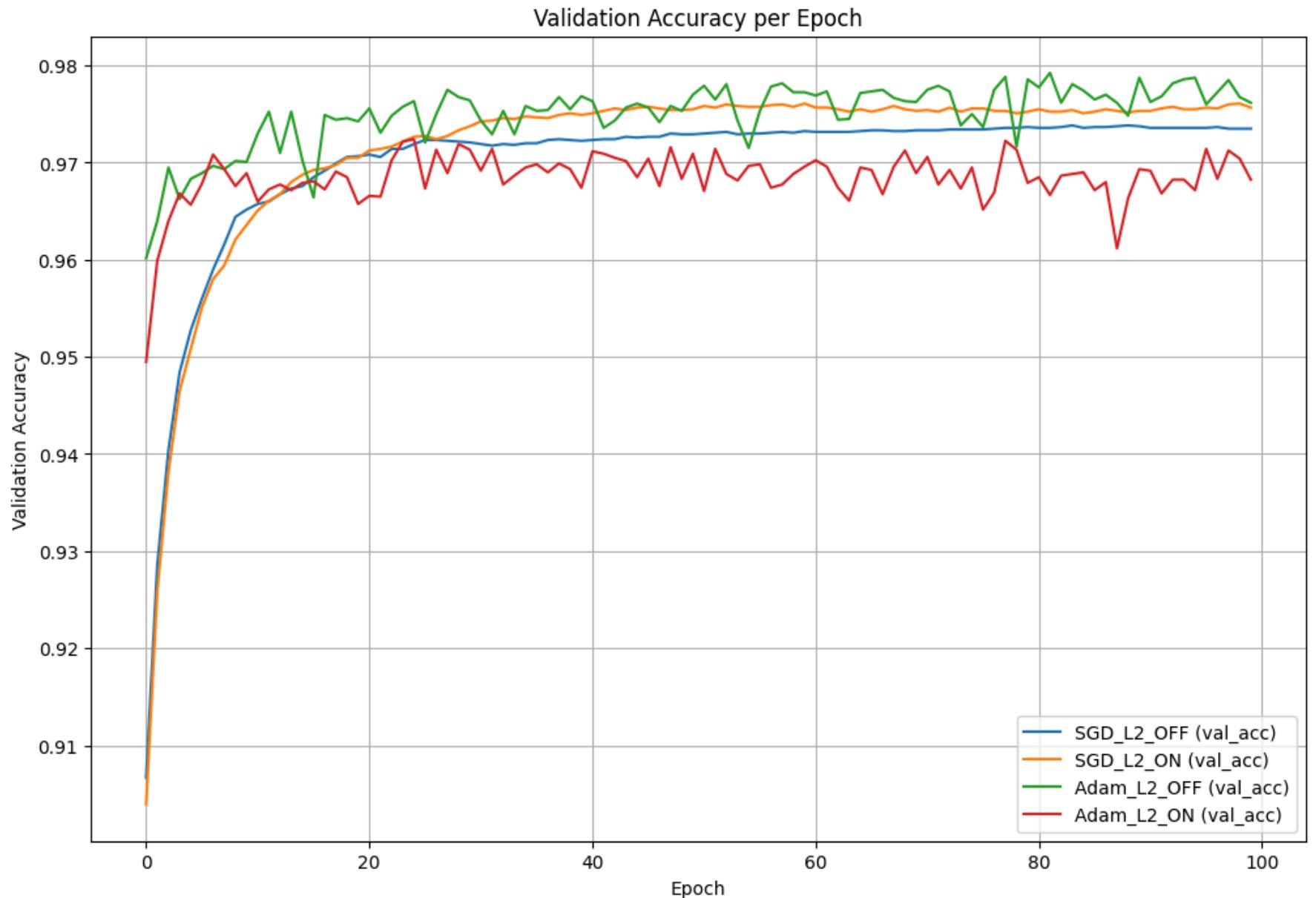
Vantagens e Desvantagens

- **Vantagens:**
 - Convergência mais rápida em problemas não estacionários.
 - Requer menos ajuste manual da taxa de aprendizado.
- **Desvantagens:**
 - Pode levar a soluções subótimas em alguns problemas.
 - Mais intensivo computacionalmente do que SGD.



Done. Test Accuracy: 0.9754
Training model: SGD_L2_ON...
Done. Test Accuracy: 0.9774
Training model: Adam_L2_OFF...
Done. Test Accuracy: 0.9759
Training model: Adam_L2_ON...
Done. Test Accuracy: 0.9691

✓ 42m27s conclusão: 11:09



Obrigado!

Dúvidas ou sugestões:
email@apresentadores

INF
INSTITUTO DE
INFORMÁTICA

