UNIVERSIDADE FEDERAL DO MARANHÃO
DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO
ENGENHARIA DA COMPUTAÇÃO
EECP0053 - TÓPICOS EM ENGENHARIA DA COMPUTAÇÃO II - FUNDAMENTOS DE REDES NEURAIS

- **Callbacks, Schedulers e Funções Customizadas para Ajustes e Monitoramento Automáticos no Treinamento do Modelo**

- ANTONIO FIALHO DA SILVA NETO
- JUSTINO FELIPE LOPES NUNES
- THALES GUSTAVO MENDES

CALLBACKS

INTRODUÇÃO

- Treinar redes neurais pode levar horas ou dias
- São ferramentas essenciais no treinamento de modelos de deep learning que ajudam a melhorar o desempenho, evitar problemas
- Importância da automação para evitar overfitting e reduzir custo computacional

- **Callbacks: Assistentes Inteligentes do Treinamento**
  - São funções ou classes que são chamadas em pontos específicos durante o treinamento para executar ações personalizadas

CALLBACKS

O que você pode fazer com retornos de chamada?

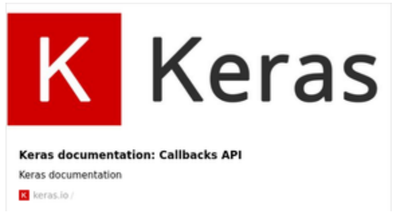
- Atualizar taxas de aprendizagem
- Visualizar gradientes
- Enviar e-mails durante o treinamento (ou após o término)
- …
- Você só é limitado pela sua imaginação

(Sebastian Raschka., 2023)

CALLBACKS KERAS

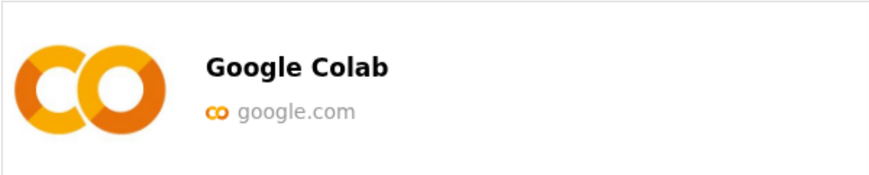
CALLBACKS KERAS:

- EarlyStopping
- ModelCheckpoint
- ReduceLROnPlateau
- TensorBoard
- LearningRateScheduler
- Base Callback class
- BackupAndRestore
- RemoteMonitor
- LambdaCallback
- TerminateOnNaN
- CSVLogger
- ProgbarLogger
- SwapEMAWeights

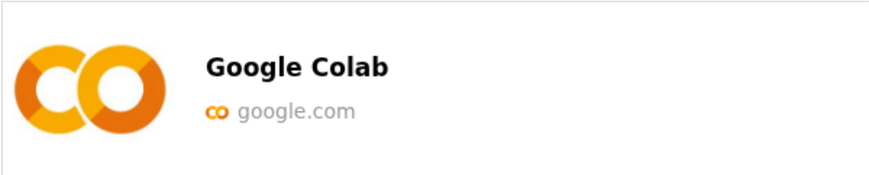


CALLBACKS

Callback Keras CSVLogger



Callback Keras EarlyStopping



CALLBACKS

Callback	Descrição/Pontos Principais
EarlyStopping	Interrompe o treinamento quando uma métrica (ex.: val_loss) para de melhorar.
ModelCheckpoint	Salva o modelo em intervalos regulares (ex.: melhor epoch ou a cada epoch).
ReduceLROnPlateau	Reduz a taxa de aprendizado quando uma métrica estagna.
TensorBoard	Salva logs para visualização no TensorBoard (gráficos, métricas, etc.).
LearningRateScheduler	Ajusta a taxa de aprendizado conforme uma função pré-definida (ex.: decay exponencial).
Base Callback class	Classe base para criar callbacks personalizados.
BackupAndRestore	Faz backup do modelo e restaura em caso de interrupção (útil para treinos longos).

CALLBACKS

RemoteMonitor	Envia logs para um servidor remoto via HTTP.
LambdaCallback	Cria callbacks simples com funções lambda (ex.: log em epoch específico).
TerminateOnNaN	Interrompe o treinamento se os gradientes/valores se tornarem NaN.
CSVLogger	Salva métricas de treinamento em um arquivo CSV.
ProgbarLogger	Exibe métricas em uma barra de progresso durante o treinamento.
SwapEMAWeights	Mantém uma média móvel exponencial (EMA) dos pesos do modelo (útil para inferência).

CALLBACKS PYTORCH

CALLBACKS PYTORCH:

- BaseCSVWriter
- GarbageCollector
- Lambda
- LearningRateMonitor
- ModuleSummary
- PyTorchProfiler
- SystemResourcesMonitor
- TensorBoardParameterMonitor
- IterationTimeLogger
- TorchSnapshotSaver
- TQDMProgressBar
- TrainProgressMonitor

https://docs.pytorch.org/tnt/stable/framework/callbacks.html

CALLBACKS

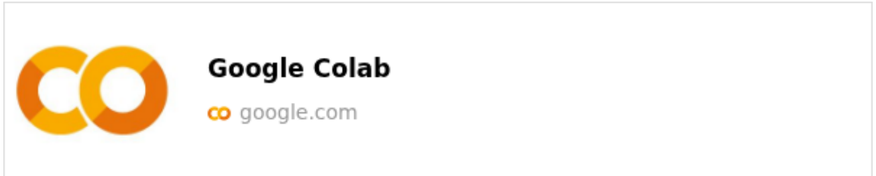
Callback	Descrição/Pontos Principais
BaseCSVWriter	Grava saídas de previsão em um arquivo CSV.
GarbageCollector	Executa coleta de lixo síncrona periodicamente.
Lambda	Executa funções personalizadas durante os loops de treino, avaliação e previsão.
LearningRateMonitor	Registra a taxa de aprendizado de otimizadores e planejadores.
ModuleSummary	Gera e registra um resumo dos módulos.
PyTorchProfiler	Cria perfis de código do usuário com PyTorch Profiler.

CALLBACKS

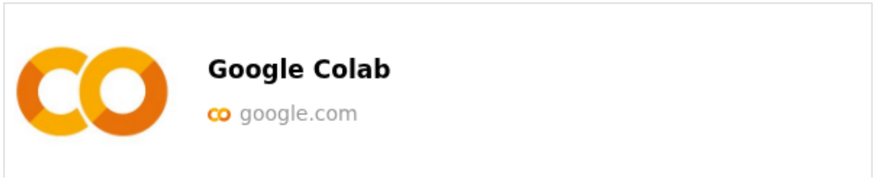
SystemResourcesMonitor	Registra uso da CPU, RAM, GPU e memória CUDA.
TensorBoardParameterMonitor	Registra parâmetros do módulo como histogramas no TensorBoard.
IterationTimeLogger	Registra tempos de iteração como escalares no TensorBoard.
TorchSnapshotSaver	Salva periodicamente o estado do app usando TorchSnapshot.
TQDMProgressBar	Mostra barra de progresso durante treino, avaliação e previsão.
TrainProgressMonitor	Registra progresso do treinamento em etapas vs épocas.

CALLBACKS

Callback pytorch BaseCSVWriter



Callback pytorch TrainProgressMonitor.



SCHEDULERS

LEARNING RATE SCHEDULERS

- Algoritmos que ajustam automaticamente a taxa de aprendizado do seu modelo durante o treinamento
- Capacidade de otimizar diferentes fases do treinamento
- Convergência mais rápida e treinamento mais estável em comparação com taxas de aprendizado fixas

LEARNING RATE SCHEDULERS

- Step Decay
- Exponential Decay
- Cosine Annealing
- Cyclical Learning Rates

STEP DECAY

- Reduz a taxa de aprendizado por um fator fixo em intervalos regulares

$$n_t = n_0 \times \gamma^{[\frac{t}{N}]}$$

Onde:

- n(t): Taxa de aprendizado na época t
- n(0) : Taxa de aprendizado inicial
- γ: Fator de decaimento
- N : Número épocas até a próxima redução
- t : Época atual

STEP DECAY

n(0) = 0,1; γ = 0,1; N = 30

Época (t)	Cálculo	n(t)
0	$0,1 \times 0,1^{0}$	0,1
1	$0,1 \times 0,1^{0}$	0,1
30	$0,1 \times 0,1^{1}$	0,01
60	$0,1 \times 0,1^{2}$	0,001

EXPONENTIAL DECAY

- Reduz suavemente a taxa de aprendizado multiplicando-a por um fator de decaimento a cada época

$$n_t = n_0 \times \gamma^t$$

Onde:

- n(t): Taxa de aprendizado na época t
- n(0): Taxa de aprendizado inicial
- y: Fator de decaimento
- t : Época atual

EXPONENTIAL DECAY

n(0) = 0,1; y = 0,5

Época (t)	Cálculo	n(t)
0	$0,1 \times 0,5^{0}$	0,1
1	$0,1 \times 0,5^{1}$	0,05
2	$0,1 \times 0,5^{2}$	0,025
3	$0,1 \times 0,5^{3}$	0,0125

COSINE ANNEALING

- Segue uma curva de cosseno, começando em um valor alto e diminuindo suavemente até um valor mínimo

$$n_t = n_{min} + 0,5(n_{max} - n_{min})(1 + cos(\frac{t}{T}\pi))$$

Onde:

- n(max) : Taxa de aprendizado máxima
- n(min) : Taxa de aprendizado mínima
- T : Número total de Épocas
- t : Época atual

COSINE ANNEALING

n(min) = 0,001; n(max) = 0,1; T = 100

Época (t)	$cos(\frac{t}{T}\pi)$	Cálculo	n(t)
0	$cos(0) = 1$	$0,001 + 0,5 \times 0,099 \times 2$	0,1
50	$cos(\pi/2) = 0$	$0,001 + 0,5 \times 0,099 \times 1$	0,0505
100	$cos(\pi) = -1$	$0,001 + 0,5 \times 0,099 \times 0$	0,001

CYCLICAL LEARNING RATES

- Oscila entre taxas de aprendizado mínimas e máximas em um padrão

$$n_t = n_{min} + (n_{max} - n_{min}) \times max(0, (1 - |x|))$$

$$\text{Para } x = \frac{t}{m} - 2 \times (\frac{1+t}{2 \times m})$$

Onde:

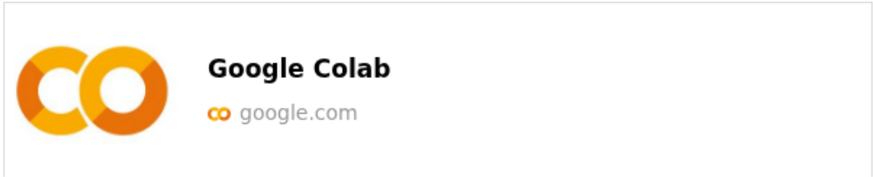
- n(max) : Taxa de aprendizado máxima.
- n(min) : Taxa de aprendizado mínima.
- m = Número de iterações para metade do ciclo.
- t : Época atual.

CYCLICAL LEARNING RATE

n(min) = 0,001; n(max) = 0,1; T = 20

Época	Cálculo de x	1 - x	Cálculo de n(t)	n(t)
0	1	0	$n_t = 0,001 + (0,1 - 0,001) \times 0)$	0,001
20	0	1	$n_t = 0,001 + (0,1 - 0,001) \times 1)$	0,1
40	1	0	$n_t = 0,001 + (0,1 - 0,001) \times 0)$	0,001

LEARNING RATE SCHEDULERS



Funções Customizadas em Machine Learning e Deep Learning

O que são Funções Customizadas?

- Funções customizadas são blocos de código que nos dão o poder de ir além do que as bibliotecas e *frameworks* (como TensorFlow, PyTorch, Keras) oferecem "de fábrica"
- Os principais componentes onde aplicamos customização são:
    a. Funções de Custo (Loss Functions)
    b. Funções de Métrica
    c. Funções de Ajuste Dinâmico de Parâmetros (Learning Rate Schedulers)
    d. Callbacks Personalizados

Por que Usar Funções Customizadas?

- Correspondência com o Objetivo de Negócio ou Pesquisa
- Lidar com a Complexidade dos Dados e do Problema
- Controle Granular sobre o Processo de Treinamento

Tipos Comuns de Funções Customizadas

Funções de Perda Personalizadas (Custom Loss Functions)

- As funções de perda são o compasso do seu modelo. Elas fornecem o sinal de erro que o algoritmo de otimização (como Descida de Gradiente) usa para ajustar os pesos
- Função de perda que penaliza mais os erros positivos (y>ŷ) que os negativos

$$\text{Loss} = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} \alpha \cdot (y_i - \hat{y}_i)^2, & \text{se } y_i > \hat{y}_i \\ (y_i - \hat{y}_i)^2, & \text{caso contrário} \end{cases}$$

- N é o número de amostras no *batch*
- yi é o valor verdadeiro para a i-ésima amostra
- ŷi é a previsão para a i-ésima amostra
- α é o fator de penalidade para subestimação ou erros positivos

Por que customizar a função de perda?

- Para Ditar o Que o Modelo Deve Aprender
  - Em detecção de câncer: penalizar mais os falsos negativos do que os falsos positivos.
  - Em previsão de falência de banco: errar ao prever que tudo vai bem pode ser muito custoso.
- Para Lidar com o Desequilíbrio de Classes
  - Em detecção de fraude bancária: fraudes são raras, mas importantes.
  - Em reconhecimento facial: ajustar perda para dar peso igual a todas as etnias ou gêneros.
- Para Incorporar Conhecimento de Domínio na Otimização
  - Em manutenção preditiva: evitar que o modelo subestime falhas que custam caro.
  - Em classificação médica: usar penalidades mais altas para erros em pacientes de risco.
- Para Implementar Novas Regularizações
  - Em redes neurais profundas: adicionar perda de regularização L1 ou L2 para evitar overfitting.
  - Em modelos de séries temporais: incentivar suavidade nas previsões com termos de penalização.

Métricas Personalizadas (Custom Metrics)

- Métricas são a forma como medimos o sucesso do nosso modelo na linguagem do problema. Elas não são usadas para otimizar, mas para entender

- O **F1-Score** é particularmente útil quando temos um desequilíbrio significativo entre as classes ou quando a importância de falsos positivos e falsos negativos é diferente, mas equilibrada

$$\text{Precision (P)} = \frac{TP}{TP + FP}$$

$$\text{Recall (Sensibility, R)} = \frac{TP}{TP + FN}$$

$$\text{F1-Score} = 2 \cdot \frac{P \cdot R}{P + R}$$

Callbacks Personalizados

Callbacks são como agentes de monitoramento e controle que você pode implantar em diferentes fases do treinamento. Eles te dão a flexibilidade de intervir no processo sem ter que modificar o loop de treinamento principal

Por que customizar os callbacks?

- Para Monitorar e Intervir no Processo de Treinamento em Tempo Real
- Para Automatizar Ações Essenciais
- Para Implementar Lógicas de Treinamento Avançadas e Dinâmicas
- Para Integrar Ferramentas Externas

Exemplos