Lab. 06: Consertando o Modelo com Dropout e Early Stopping

Aléssio Miranda Júnior

Agosto - 2026/2

Lab. 06: Regularização e Dropout

Capítulo em Revisão

Este conteúdo está sendo revisado. Algumas informações podem estar preliminares, conter erros de formatação ou sofrer alterações na versão final.

⋆ Objetivo do Laboratório

No Lab 05, você provocou Overfitting deliberadamente e viu a “boca de jacaré” entre as curvas de treino e validação. Hoje você vai **curar** aquele modelo doente. Usando o **mesmo dataset**, aplicará três técnicas profissionais — `Dropout`, `Regularização L2` e `EarlyStopping` — e comparará visualmente os gráficos **Antes vs Depois**. Ao final, terá dominado a **caixa de ferramentas completa** contra o Overfitting, a mesma usada em produção por engenheiros do Google e da Meta.

CAPÍTULO EM REVISÃO (24 de agosto de 2026) O conteúdo pode sofrer alterações.

Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

1 Parte 1: O Problema F — Estilo Maratona

Problema F: A Clínica do Modelo Doente

Contexto: No Lab 05, o modelo da *SmartFactory* atingiu 99% de acurácia no treino, mas apenas ≈75% na validação — ele "decorou" os dados em vez de aprender padrões. O CTO da empresa está furioso: *"Gastamos milhares em GPU para um modelo que não funciona em produção!"*. Sua missão é **curar** o modelo usando a caixa de ferramentas de regularização.

Condições do Experimento:

- **Dataset:** Idêntico ao Lab 05 (500 amostras, 10 features, `flip_y=0.1`)
- **Modelo doente (controle):** Rede 256-128-64 sem proteção, 300 épocas
- **Modelo curado (tratamento):** Rede 128-64-32 com Dropout + L2 + Early Stopping

Modelo de Saída: Gráfico comparativo **Antes vs Depois** mostrando:

- **Antes:** Curvas divergentes (boca de jacaré) — Overfitting
- **Depois:** Curvas paralelas (caminham juntas) — Modelo saudável

Critérios de Aprovação:

1. **Early Stopping** ativou antes de 500 épocas
2. **Gap** (loss validação − loss treino) do modelo regularizado < gap do modelo sem proteção
3. Previsões do modelo no intervalo válido $[0, 1]$
4. `restore_best_weights = True` configurado no callback

2 Parte 2: Recriando o Dataset do Lab 05 (5 min)

Crie o arquivo `src/modelo_regularizado.py`. Reutilizaremos o mesmo dataset do Lab 05 para garantir comparação justa:

```
import numpy as np
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import matplotlib
matplotlib.use('Agg')
```

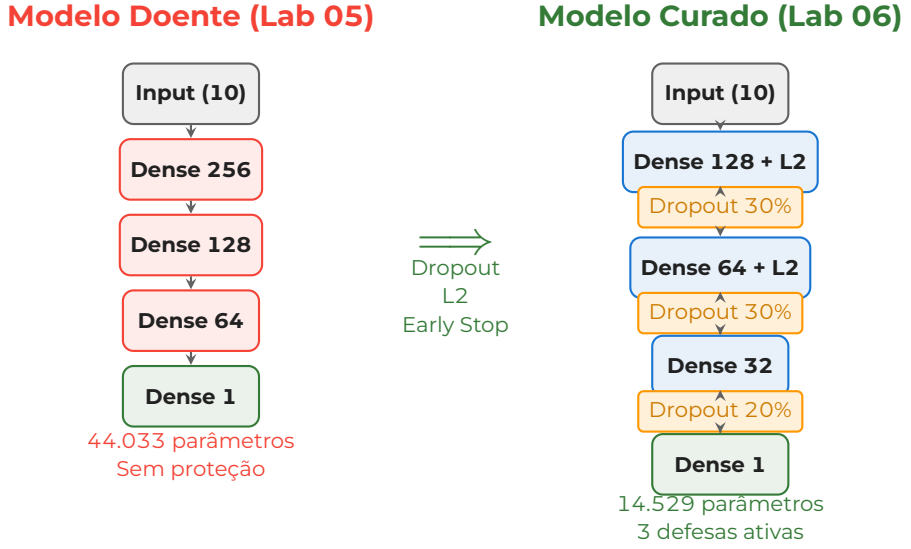
Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.regularizers import l2

# Mesmo dataset do Lab 05 (reprodutibilidade garantida)
X, y = make_classification(
    n_samples=500, n_features=10,
    n_informative=5, n_redundant=3,
    random_state=42, flip_y=0.1
)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_norm = scaler.fit_transform(X_train)
X_test_norm = scaler.transform(X_test)
```

3 Parte 3: Diagnóstico Visual — Rede Doente vs Curada

Antes de codar, observe o diagrama que compara as duas arquiteturas. A chave está nos blocos de Dropout (representados como “filtros” que desligam neurônios) e na penalidade L2 nos pesos:



Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

4 Parte 4: O Modelo Curado com Dropout + L2 (10 min)

Construa o modelo regularizado. Observe as três camadas de defesa:

```python
def criar_modelo_regularizado():
    """Rede com Dropout + L2 como vacina contra Overfitting."""
    model = Sequential()
    # L2 penaliza pesos grandes na loss
    model.add(Dense(128, activation='relu', input_shape=(10,),
                    kernel_regularizer=l2(0.001)))
    model.add(Dropout(0.3))    # Desliga 30% dos neurônios

    model.add(Dense(64, activation='relu',
                    kernel_regularizer=l2(0.001)))
    model.add(Dropout(0.3))

    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.2))    # Menos agressivo perto da saída

    model.add(Dense(1, activation='sigmoid'))

    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
    return model
```

• Teoria: As Três Defesas Combinadas

Técnica	Mecanismo	Analogia
Dropout (0.3)	Desliga 30% dos neurônios a cada batch	"Rodízio de funcionários"
L2 (λ=0.001)	Penaliza $\sum w_i^2$ na loss	"Imposto sobre pesos grandes"
Early Stopping	Para o treino quando val_loss sobe	"Sentinela automática"

Nenhuma técnica sozinha resolve todos os casos. Na prática, **combinamos as três** — assim como um médico prescreve múltiplos tratamentos para uma doença resistente.

5 Parte 5: Configurando o Early Stopping (8 min)

Configure o callback que monitora `val_loss` e para o treino automaticamente:

```python
# O "vigia" que monitora a validação
vigilante = EarlyStopping(
    monitor='val_loss',          # Métrica monitorada
```

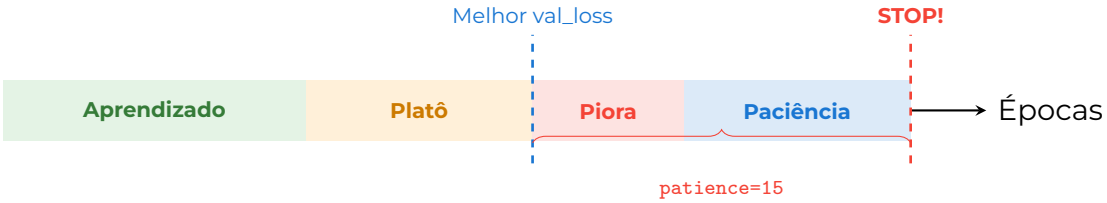
Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.

```
    patience=15,                # Tolerância: 15 épocas sem melhora
    restore_best_weights=True   # Restaura os pesos do vale!
)
```

△ Importante: O Erro de Esquecer restore_best_weights

Sem `restore_best_weights=True`, o Keras para o treino mas mantém os **últimos** pesos (que já estão contaminados pelo início do Overfitting). Com a flag ativa, ele "viaja no tempo" e restaura os pesos do momento ótimo — a época onde `val_loss` atingiu o mínimo global.

A. Linha do Tempo do Early Stopping



6 Parte 6: Treinamento Comparativo (12 min)

Treine ambos os modelos (doente e curado) com o mesmo dataset para comparação justa:

```
# === MODELO CURADO (com proteção) ===
modelo_reg = criar_modelo_regularizado()
hist_reg = modelo_reg.fit(
    X_train_norm, y_train,
    epochs=500,              # Teto alto (Early Stopping decide)
    batch_size=16,
    validation_split=0.2,
    callbacks=[vigilante],
    verbose=0
)

# === MODELO DOENTE (controle, sem proteção) ===
def criar_modelo_sem_protecao():
    """Rede do Lab 05 sem nenhuma defesa."""
    model = Sequential()
    model.add(Dense(256, activation='relu', input_shape=(10,)))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

```

Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

```
modelo_sem = criar_modelo_sem_protecao()
hist_sem = modelo_sem.fit(
    X_train_norm, y_train,
    epochs=300, batch_size=16,
    validation_split=0.2, verbose=0
)
```

Agora gere o gráfico comparativo lado a lado:

```
def plotar_comparacao(hist_sem, hist_reg):
    """Compara Antes (overfitting) vs Depois (regularizado)."""
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

    # Gráfico 1: SEM proteção
    ax1.plot(hist_sem.history['loss'], 'b-',
             label='Loss Treino')
    ax1.plot(hist_sem.history['val_loss'], 'r-',
             label='Val Loss')
    ax1.set_title('SEM Regularização (Overfitting)')
    ax1.set_xlabel('Épocas')
    ax1.set_ylabel('Loss')
    ax1.legend()
    ax1.grid(True, alpha=0.3)

    # Gráfico 2: COM Dropout + L2 + Early Stopping
    ax2.plot(hist_reg.history['loss'], 'b-',
             label='Loss Treino')
    ax2.plot(hist_reg.history['val_loss'], 'r-',
             label='Val Loss')
    ax2.set_title(f'COM Regularização (parou na época '
                  f'{len(hist_reg.epoch)})')
    ax2.set_xlabel('Épocas')
    ax2.set_ylabel('Loss')
    ax2.legend()
    ax2.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.savefig('modelo_consertado.png', dpi=150)
    plt.close()
    print("Gráfico salvo: modelo_consertado.png")
```

B. Tabela Comparativa Esperada

Ao rodar o script, você deve observar números semelhantes a estes:

Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

Métrica	Sem Proteção (Lab 05)	Com Regularização (Lab 06)
Épocas executadas	300 (todas)	≈60–120 (Early Stopping)
Acurácia no treino	≈99%	≈82–88%
Acurácia na validação	≈75–80%	≈80–85%
Gap (treino − val)	≈20%	≈2–5%
Loss treino final	≈0.02	≈0.35
Loss validação final	≈0.70	≈0.40
Tempo de treino	Longo (300 épocas)	Curto (Early Stop)

Observe o paradoxo: o modelo regularizado tem acurácia **menor** no treino (não decora!), mas acurácia **maior** na validação (generaliza!). Isso é exatamente o que queremos em produção.

7 Parte 7: Bloco Principal e Execução (5 min)

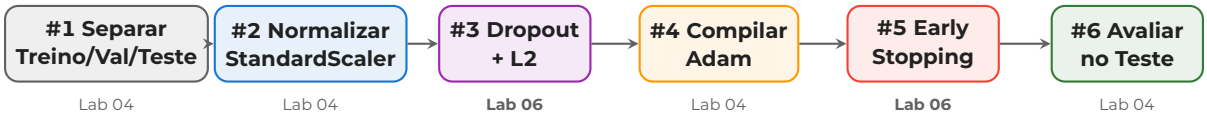
Monte o bloco principal com o relatório completo:

```python
if __name__ == "__main__":
    plotar_comparacao(hist_sem, hist_reg)

    epocas_exec = len(hist_reg.epoch)
    _, acc_reg = modelo_reg.evaluate(
        X_test_norm, y_test, verbose=0)
    _, acc_sem = modelo_sem.evaluate(
        X_test_norm, y_test, verbose=0)

    print(f"\n{'='*50}")
    print(f"  MODELO SEM PROTEÇÃO (Lab 05):")
    print(f"    Épocas: 300 (todas)")
    print(f"    Acc Teste: {acc_sem:.2%}")
    print(f"")
    print(f"  MODELO REGULARIZADO (Lab 06):")
    print(f"    Épocas: {epocas_exec} (Early Stopping)")
    print(f"    Acc Teste: {acc_reg:.2%}")
    print(f"{'='*50}")
```

Execute `python src/modelo_regularizado.py` e verifique que o Early Stopping parou antes de 500 épocas e que as curvas do gráfico caminham juntas.

8 Parte 8: A Receita Profissional (Referência)

Ao completar este lab, você dominou todos os passos da receita profissional de Machine Learning:



Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

Esta receita é válida para **qualquer** problema de classificação com dados tabulares. É a base que você usará no **TP1** do módulo.

9 Parte 9: Garantia de Qualidade — Testes CI/CD (5 min)

Os testes já estão prontos no arquivo `tests/test_regularizado.py` do seu repositório:

```python
import numpy as np
from src.modelo_regularizado import (
    criar_modelo_regularizado, criar_modelo_sem_protecao,
    X_train_norm, y_train, X_test_norm, y_test,
    hist_reg, hist_sem, modelo_reg, vigilante
)

def test_early_stopping_ativou():
    """Early Stopping DEVE ter parado antes de 500."""
    epocas = len(hist_reg.epoch)
    assert epocas < 500, \
        f"Early Stopping falhou: {epocas} épocas"

def test_gap_reduzido():
    """Gap treino-validação menor com regularização."""
    loss_reg = hist_reg.history['loss'][-1]
    val_loss_reg = hist_reg.history['val_loss'][-1]
    gap_reg = abs(val_loss_reg - loss_reg)

    loss_sem = hist_sem.history['loss'][-1]
    val_loss_sem = hist_sem.history['val_loss'][-1]
    gap_sem = abs(val_loss_sem - loss_sem)

    assert gap_reg < gap_sem, \
        f"Gap não reduziu: {gap_reg:.4f} >= {gap_sem:.4f}"

def test_modelo_funcional():
    """Previsões devem estar entre 0 e 1."""
    preds = modelo_reg.predict(X_test_norm, verbose=0)
    assert preds.shape == (len(X_test_norm), 1)
    assert np.all(preds >= 0) and np.all(preds <= 1)

def test_callback_configurado():
    """restore_best_weights DEVE estar ativo."""
    assert vigilante.restore_best_weights == True
```

1. Rode os testes no terminal: `pytest tests/test_regularizado.py -v`.
2. Se a tela exibir **4 passed** em verde, sua implementação está correta.

Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

10 Parte 10: Commit e Push (5 min)

Com todos os testes passando localmente, efetue o envio para o GitLab:

```
git add src/modelo_regularizado.py modelo_consertado.png
git commit -m "Lab 06: Dropout + L2 + Early Stopping vs Overfitting"
git push origin main
```

O GitLab CI interceptará seu push e executará o `pytest` automaticamente. O **Selo Verde** confirma sua pontuação.

△ Importante: Se o Pipeline Falhar

Os erros mais comuns são: (1) esquecer o import `from tensorflow.keras.regularizers import l2`, (2) não passar `callbacks=[vigilante]` no `fit()`, (3) esquecer `restore_best_weights=True` no EarlyStopping. O CI roda os **mesmos testes** que você rodou localmente.

Conclusão: O que Você Construiu Hoje

✔ Takeaways

- Você aplicou **Dropout (30%)** entre camadas densas, impedindo que neurônios se co-adaptem e memorizem ruído — cada neurônio agora é "independente" e útil sozinho.
- Adicionou **Regularização L2** ($\lambda = 0.001$) que penaliza pesos grandes na função de perda: $\mathcal{L}_{total} = \mathcal{L}_{dados} + \lambda \sum w_i^2$.
- Configurou **Early Stopping** com `patience=15` e `restore_best_weights=True`, automatizando a parada ótima e restaurando os pesos do melhor momento.
- Comparou visualmente o gráfico **Antes (Overfitting)** vs **Depois (Regularizado)**, comprovando que as curvas agora caminham juntas e o gap é mínimo.
- Dominou a **receita profissional de 6 passos** para treinar qualquer modelo de classificação tabular robusto.

Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

◇ **Informação**

Gancho para a Próxima Aula: Até agora, todos os seus modelos processaram **tabelas numéricas** (X com linhas e colunas). Mas a IA do mundo real precisa enxergar **imagens**: fotos, radiografias, satélites. Uma imagem 256×256 RGB tem $256 \times 256 \times 3 = 196.608$ pixels — se aplicássemos um `Dense` diretamente, teríamos **milhões de parâmetros** e Overfitting instantâneo. Na próxima aula, entraremos no universo das **Redes Convolucionais (CNNs)**, que exploram a *estrutura espacial* dos pixels para aprender com ordens de grandeza menos parâmetros, e do **Transfer Learning** — onde aproveitamos redes pré-treinadas por Google e Meta para resolver problemas com *poucos dados*.

Versão compilada em 24 de agosto de 2026 – Capítulo isolado do livro.