Documentação Técnica de Desenvolvimento de Software

1. Título do Projeto: Modelo de Deep Learning fast.ai para Detecção de Neoplasias Cutâneas

2. Versão: 1.0

3. Data: 09 de junho de 2025

4. Autores: Rodrigo Aglinskas

5. Introdução

Este documento descreve o desenvolvimento de um modelo de deep learning utilizando a biblioteca fastai para a classificação e detecção de neoplasias cutâneas. O objetivo principal é auxiliar profissionais de saúde no diagnóstico precoce e preciso de lesões dermatológicas, contribuindo para a melhoria dos resultados dos pacientes. O modelo foi treinado com o dataset HAM10000.

6. Arquitetura do Software

O software é construído como um notebook Jupyter interativo, o que facilita a experimentação, visualização e o fluxo de trabalho de machine learning. A arquitetura segue as melhores práticas de deep learning, utilizando uma rede neural convolucional (CNN) pré-treinada (ResNet18) e a metodologia fastai para otimização e treinamento.

7. Componentes Principais

- **Google Colab:** Ambiente de desenvolvimento utilizado, permitindo acesso a GPUs para treinamento acelerado.
- **Google Drive:** Utilizado para armazenamento do dataset e do modelo treinado.

- **Fastai:** Biblioteca de deep learning de alto nível construída sobre PyTorch, simplificando o processo de treinamento e implantação de modelos.
- **Pandas:** Para manipulação e análise de dados tabulares (metadados do HAM10000).
- **Matplotlib e Seaborn:** Para visualização e análise exploratória dos dados.
- **Numpy:** Para operações numéricas.
- **Dataset HAM10000:** Um dataset público de imagens de lesões de pele com sete classes diagnósticas, crucial para o treinamento e validação do modelo.

8. Requisitos de Sistema

- **Ambiente de Execução:** Google Colab com GPU.
- **Bibliotecas Python:**
    - fastai
    - pandas
    - matplotlib
    - numpy
    - seaborn
    - scikit-learn (para métricas e auxiliares)
    - Pillow (dependência do fastai/torchvision)
- **Armazenamento:** Espaço no Google Drive para o dataset HAM10000 (aproximadamente 3 GB de imagens e o arquivo CSV de metadados).
- **Hardware (para treinamento):** GPU (essencial para treinamento eficiente de modelos de deep learning).

9. Instalação e Configuração

1. **Acessar Google Colab:** Abrir um novo notebook no Google Colab.
2. **Montar Google Drive:**

   ```
   from google.colab import drive

   drive.mount('/content/drive')
   ```

3. **Organizar Dados:** Certificar-se de que o dataset HAM10000 (imagens e HAM10000_metadata.csv) esteja na pasta /content/drive/MyDrive/archive ou ajustar o base_skin_dir e csv_path no código para o local correto do dataset. O dataset pode ser baixado do Kaggle.
4. **Instalar Fastai:**

   ```
   !pip install fastai
   ```

5. **Carregar Bibliotecas:** Assegurar que todas as bibliotecas necessárias sejam importadas no início do notebook.

   ```
   # Carregar toda vez que acionar a biblioteca FASTAI

   # Recarregar notebook quaisquer alterações feitas em qualquer biblioteca usada.

   %reload_ext autoreload

   %autoreload 2


   # Garantir que todos os gráficos plotados sejam mostrados

   %matplotlib inline


   # Carga bibliotecas

   from fastai.vision.all import *

   from fastai.metrics import * # Pode ser necessário ajustar se apenas Accuracy for usada
   ```

```
import pandas as pd
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
import os
from glob import glob
import seaborn as sns
```

10. Estrutura de Pastas

```
/content/drive/MyDrive/
└── archive/
├── HAM10000_metadata.csv
├── HAM10000_images_part1/
├── HAM10000_images_part2/
└── ... (outras partes do dataset)
```

11. Fluxo de Desenvolvimento e Execução

1. **Montagem do Drive e Configuração Inicial:** Seção de Instalação e Configuração.
2. **Carregamento de Dados:**
    - Leitura do arquivo HAM10000_metadata.csv para um DataFrame Pandas.
    - Combinação do DataFrame com os caminhos das imagens.
3. **Análise Exploratória de Dados (EDA):**

- Visualização da distribuição de classes (dx), idade (age), sexo (sex) e localização (localization).
- Exibição de amostras de imagens por tipo de lesão.

4. **Pré-processamento e Aumento de Dados:**
   - Criação de DataLoaders da fastai, que gerenciam o carregamento, redimensionamento e transformações das imagens.
   - Aplicação de técnicas de aumento de dados (flips, rotações, zoom, etc.) para melhorar a robustez do modelo.
5. **Treinamento do Modelo:**
   - Definição de uma arquitetura de modelo (ex., resnet18).
   - Utilização da função cnn_learner da fastai para criar o Learner (responsável pelo treinamento).
   - Configuração da função de perda (ex., CrossEntropyLossFlat).
   - Treinamento do modelo usando fit_one_cycle ou métodos semelhantes.
   - Uso de fine_tune para transfer learning eficaz.
6. **Avaliação do Modelo:**
   - Cálculo de métricas de desempenho (ex., accuracy).
   - Geração de matriz de confusão para analisar o desempenho por classe.
   - Visualização de exemplos onde o modelo acertou e errou.
7. **Interpretabilidade:**
   - Uso de ClassificationInterpretation para entender as predições do modelo.
8. **Exportação e Implantação:**
   - Exportação do modelo treinado para um arquivo .pkl para uso posterior.

12. Funções e Módulos Chave

- **google.colab.drive.mount():** Monta o Google Drive.
- **os.path.join():** Constrói caminhos de arquivo.
- **pd.read_csv():** Lê o arquivo de metadados.
- **skin_df.sort_values():** Ordena o DataFrame.
- **ImageDataLoaders.from_df():** Cria DataLoaders a partir de um DataFrame e caminhos de imagem.
- **cnn_learner():** Inicializa o modelo de deep learning com uma CNN pré-treinada.
- **learn.fine_tune():** Realiza o treinamento fino (fine-tuning) do modelo.
- **learn.export():** Salva o modelo treinado.
- **learn.predict():** Realiza inferência em novas imagens.
- **ClassificationInterpretation.from_learner():** Cria um objeto para interpretar os resultados do modelo.
- **interp.plot_confusion_matrix():** Plota a matriz de confusão.
- **interp.plot_top_losses():** Plota as previsões com maior perda.

13. Testes

- **Testes de Unidade:** Funções auxiliares e de pré-processamento são testadas isoladamente.
- **Testes de Integração:** O fluxo completo de treinamento e avaliação dentro do notebook serve como um teste de integração.
- **Testes de Desempenho:** Acompanhamento de métricas como acurácia, precisão, recall e F1-score durante o treinamento e na validação.
- **Validação Cruzada:** Aplicação de validação cruzada (k-fold) para uma avaliação mais robusta.

14. Problemas Conhecidos e Limitações

- **Viés do Dataset:** O desempenho do modelo é limitado pela qualidade e diversidade do dataset HAM10000. Desequilíbrios de classes ou representatividade limitada podem afetar a generalização.
- **Dependência de Hardware:** O treinamento requer uma GPU, o que pode ser uma barreira para usuários sem acesso a ambientes como o Google Colab Pro.
- **Complexidade da Interpretação Clínica:** Embora o modelo forneça uma classificação, a interpretação clínica final e o diagnóstico devem ser feitos por um profissional de saúde qualificado. O modelo é uma ferramenta de apoio.
- **Configuração de Caminhos:** O usuário deve garantir que os caminhos para o dataset estejam corretos no Google Drive.

15. Futuras Melhorias

- **Expansão do Dataset:** Incorporar mais dados, especialmente de classes raras, para melhorar a robustez e a generalização do modelo.
- **Técnicas de Balanceamento de Classes:** Implementar estratégias como oversampling, undersampling ou focal loss para lidar com desequilíbrios de classes no dataset.
- **Interface de Usuário:** Desenvolver uma interface gráfica do usuário (GUI) ou uma API web para tornar o modelo mais acessível a não-desenvolvedores.
- **Implantação em Nuvem:** Implantar o modelo em plataformas de nuvem (AWS, GCP, Azure) para escalabilidade e disponibilidade.
- **Métricas Adicionais:** Explorar outras métricas de avaliação, como AUC-ROC, para uma análise mais completa.
- **Ensemble Learning:** Combinar múltiplos modelos para melhorar o desempenho geral.

- **Otimização de Hiperparâmetros:** Utilizar ferramentas como Weights & Biases ou Optuna para otimizar os hiperparâmetros do modelo de forma sistemática.

16. Referências

AHEDJNEED. Skin cancer classifier with fastai (acc: 97). Disponível em: https://www.kaggle.com/code/ahedjneed/skin-cancer-classifier-with-fastai-acc-97. Acesso em: 27 out. 2023.

LEONBLUM. HAM10000 vision (ResNet18) - 97.7% accuracy. Disponível em: https://www.kaggle.com/code/leonblum/ham10000-vision-resnet18-97-7-accuracy/notebook. Acesso em: 27 out. 2023.

RKUO2000. Skin lesion classification. Disponível em: https://www.kaggle.com/code/rkuo2000/skin-lesion-classification. Acesso em: 27 out. 2023.

KMADER. Skin cancer MNIST HAM10000. Disponível em: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000/code?resource=download. Acesso em: 27 out. 2023.

RSLU2000. Skin cancer model - 97.88% accuracy. Disponível em: https://www.kaggle.com/code/rslu2000/skin-cancer-model-97-88-accuracy. Acesso em: 27 out. 2023.

KMADER. Dermatology MNIST: loading and processing. Disponível em: https://www.kaggle.com/code/kmader/dermatology-mnist-loading-and-processing. Acesso em: 27 out. 2023.

DHruV1234. HAM10000 skin disease classification. Disponível em: https://www.kaggle.com/code/dhruv1234/ham10000-skin-disease-classification. Acesso em: 27 out. 2023.

17. Glossário

- **CNN (Convolutional Neural Network):** Rede Neural Convolucional, tipo de rede neural ideal para processamento de imagens.
- **Deep Learning:** Subcampo do Machine Learning que utiliza redes neurais com múltiplas camadas.
- **Fastai:** Biblioteca Python de deep learning que simplifica o treinamento de modelos.
- **HAM10000:** Dataset de imagens de lesões de pele.
- **Jupyter Notebook:** Ambiente de computação interativa que permite combinar código, texto e visualizações.
- **Transfer Learning:** Técnica de machine learning onde um modelo pré-treinado em uma tarefa é reutilizado como ponto de partida para uma nova tarefa.
- **Aumento de Dados (Data Augmentation):** Técnicas usadas para aumentar a quantidade e diversidade de dados de treinamento, gerando novas amostras a partir das existentes (ex., rotações, flips).
- **Matriz de Confusão:** Tabela que resume o desempenho de um algoritmo de classificação.