Relatório - Trabalho Final de Fundamentos de Big Data

1. Como funciona a Importação e Organização

A parte mais inteligente desse código não é a IA, mas sim a **Engenharia de Dados** feita nas linhas 20 a 65.

- **O Problema:** O dataset HAM 10000 original vem "bagunçado" para inteligência artificial. Ele tem uma pasta com 10.000 imagens misturadas e um arquivo CSV dizendo quem é quem. O Keras (biblioteca de IA) prefere que as imagens já estejam separadas em pastas por categoria.
- **A Solução (O Script):**
    - **Baixa:** O `kagglehub` puxa os arquivos brutos.
    - **Define a Regra de Negócio (O "Pulo do Gato"):**
        - O código não tenta classificar entre 7 doenças (o que é muito difícil). Ele simplifica o problema para **Binário** (Sim/Não).
        - Ele define que `['mel', 'bcc', 'akiec']` são **PERIGOSOS** (Câncer maligno ou pré-maligno). Todo o resto é **BENIGNO**.
    - **Reorganização Física:** O código cria pastas físicas (`./train/perigo`, `./train/benigno`) e **copia** os arquivos JPG para dentro delas baseando-se no CSV.
    - *Por que isso importa:* Isso transforma um problema complexo de 7 variáveis em um problema de "Triagem" (Vai para biópsia ou não?), aumentando drasticamente a chance de acerto.

---

2. Como funcionou a Normalização (`ImageDataGenerator`)

Nas linhas 70 a 89, ocorre o pré-processamento. A rede neural não "vê" fotos, ela vê matrizes matemáticas de números.

- **Normalização (`rescale=1./255`):**
    - Uma imagem digital tem pixels com valores de cor que variam de **0 (preto) a 255 (branco)**.
    - Redes Neurais funcionam mal com números grandes (como 255). Elas preferem números entre **0 e 1**.
    - O código divide todos os pixels por 255. Assim, o branco vira `1.0` e o preto `0.0`. Isso acelera o cálculo matemático.
- **Data Augmentation (Apenas no Treino):**

- O código aplica `rotation_range=20`, `zoom_range=0.2`, `horizontal_flip=True`.
- **O que é:** Ele cria "cópias falsas" das imagens originais girando-as ou dando zoom.
- **Por que:** Para evitar que o modelo "decore" a foto exata (Overfitting). Ele obriga o modelo a aprender o padrão da doença, não importa se a foto está de cabeça para baixo ou mais perto.

---

3. O Modelo: Estrutura, Lógica e Motivo da Escolha

Aqui está o "cérebro" (linhas 93 a 108). Estamos usando uma técnica chamada **Transfer Learning** (Aprendizado por Transferência).

A Estrutura (Arquitetura):

1. **A Base (`MobileNetV2`):**
   - É uma rede neural criada pelo Google que já viu 14 milhões de imagens (Dataset ImageNet). Ela já sabe o que é uma "borda", uma "curva", uma "textura".
   - `include _top=False`: Nós cortamos a "cabeça" dela (que servia para classificar gatos e cachorros) e ficamos só com o extrator de características.
   - `base_model.trainable = False`: Nós **congelamos** essa parte. Não vamos treinar o que o Google já fez. Isso economiza tempo e poder computacional.
2. **A Nova Cabeça (Classificador):**
   - `GlobalAveragePooling2D`: Transforma os mapas complexos da MobileNet em um vetor simples de números (resumo da imagem).
   - `Dense(128, activation='relu')`: Uma camada densa que aprende a combinar as características para achar padrões de pele.
   - `Dropout(0.5)`: **Técnica de Esquecimento**. Desliga aleatoriamente 50% dos neurônios a cada rodada. Isso força a rede a não depender de um único caminho, tornando-a mais robusta.
   - `Dense(1, activation='sigmoid')`: A saída final. É **1 único neurônio**.
     - Se o valor for perto de 0 -> Benigno.
     - Se o valor for perto de 1 -> Perigoso.

Motivo da Escolha (Mobile Net V2):

- **Eficiência:** É uma rede "leve". O arquivo final tem ~15MB. Redes mais poderosas (ResNet, Efficient Net) geram arquivos de 100 MB +.

- **Compatibilidade:** Como vocês precisam rodar isso em um Docker e num site (Streamlite), uma rede leve garante que o site não trave e que o Docker suba rápido. É a escolha perfeita para aplicações web.

---

4. A Etapa de Treinamento (`model.fit`)

É aqui que a mágica acontece (linhas 115 a 119).

- **O Processo:** O modelo olha um lote de 32 imagens (`batch_size=32`), faz uma previsão, compara com a resposta real (Benigno ou Perigoso) e ajusta os pesos matemáticos da "Nova Cabeça" para errar menos na próxima vez.
- **O Otimizador (`Adam`):** É o algoritmo que decide *como* ajustar os pesos. O Adam é o padrão de mercado porque ajusta a velocidade de aprendizado automaticamente.
- **A Função de Perda (`binary_crossentropy`):** É a fórmula matemática usada para calcular a nota de erro, específica para problemas de Sim/Não.

---

5. Conclusões Possíveis (O que esperar do relatório)

Ao olhar os resultados que sairão agora, você deve concluir baseado nestes cenários:

- **Cenário A (Sucesso - Acurácia > 80%):**
    - *Conclusão:* "O uso de Transfer Learning com a MobileNetV2 mostrou-se eficaz para a triagem de lesões de pele. A simplificação das classes para binário permitiu alta assertividade mesmo com pouco tempo de treino."
- **Cenário B (Overfitting - Treino 95% / Validação 70%):**
    - *Conclusão:* "O modelo decorou as imagens de treino mas não generalizou bem. Isso indica que precisaríamos de mais Data Augmentation ou aumentar a taxa de Dropout."
- **Cenário C (Underfitting - Acurácia < 60%):**
    - *Conclusão:* "O modelo não conseguiu aprender padrões suficientes. Talvez as lesões sejam muito parecidas ou a MobileNetV2 seja simples demais para diferenciar esses tipos específicos."

Como seu teste anterior deu **83%**, estamos esperando o **Cenário A**. Isso valida o uso da ferramenta como um "assistente de triagem" para médicos (não para diagnóstico final, mas para alerta).