PSI3471 - Fundamentos de Sistemas Eletrônicos Inteligentes

Relatório do EP - Classificação de Tipos de Pistache

Pedro Buczinski Lages de Carvalho - 12555266

1 de julho de 2025

1 Introdução

Este trabalho tem como objetivo realizar a classificação de dois tipos de pistache — Kirmizi e Siirt — a partir de imagens disponíveis em um conjunto de dados do Kaggle com 2148 imagens. O exercício é dividido em etapas sequenciais, começando pelo carregamento e pré-processamento das imagens, que são alinhadas e redimensionadas. Em seguida, são aplicadas duas abordagens de aprendizado supervisionado para a tarefa de classificação: um algoritmo clássico de aprendizado de máquina e uma rede neural convolucional (CNN).

O enunciado determina um forte desbalanceamento entre os conjuntos de treino e teste, com apenas 200 imagens para treinamento e 1948 para teste, o que torna a tarefa mais desafiadora e permite avaliar a robustez dos modelos em cenários com dados limitados. Ao final, os desempenhos dos dois métodos são comparados com base em métricas de acurácia.

Eu escrevi e rodei todos os programas na própria plataforma do Kaggle, para facilitar a importação do dataset, além da exportação dos meus dados de output para os outros programas desenvolvidos para esse trabalho.

2 Redução e Alinhamento Vertical das Imagens

Link para notebook no Kaggle

No programa alinha, preparei as imagens do dataset de pistaches para o treinamento dos modelos, alinhando-as verticalmente e padronizando seu tamanho. Para isso, comecei utilizando o código **find_center_and_orientation**, fornecido em C++, que traduzi e adaptei para Python dentro do ambiente do Kaggle para achar o centro de massa e a orientação das imagens originais convertidas em binárias.

Em seguida, implementei a função **eliminar_linhas_pretas** para remover as bordas superiores e inferiores compostas por linhas totalmente pretas. Nessa função, primeiro as imagens são convertidas para escala de cinza (grayscale). Depois, a função encontra as linhas em que todos os pixels tem valor zero. A partir disso, detecta o intervalo central da imagem que não era completamente preto e recorta apenas essa região relevante.

Com as imagens limpas, redimensionei cada uma para 64×64 pixels utilizando a função resize do OpenCV e salvei os arquivos processados em um diretório específico no Kaggle. Para realizar o alinhamento vertical, calculei o ângulo ϑ entre a orientação da imagem original e o eixo horizontal com a função **find_center_and_orientation**. Em seguida, determinei o ângulo de rotação como 90° - ϑ, garantindo que o objeto ficasse orientado verticalmente. Usei as funções getRotationMatrix2D e warpAffine, também do OpenCV, para construir e aplicar essa rotação às imagens redimensionadas.

Finalmente, salvei as imagens alinhadas em um segundo diretório. Para garantir que o processo funcionasse corretamente, exibi algumas imagens ao longo do código nas etapas de redução e alinhamento, permitindo verificar visualmente o resultado obtido.

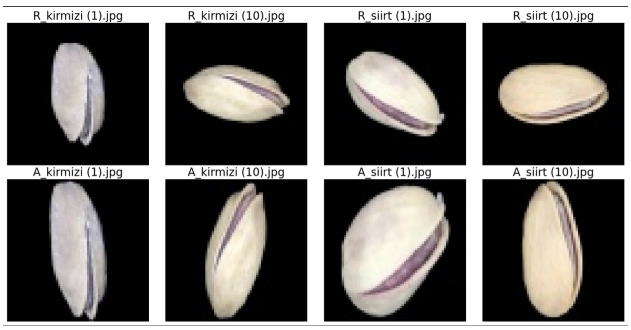

Figura 1: Imagens reduzidas e alinhadas

3 Classificação com Algoritmo Clássico (SVM)

Link para notebook no Kaggle

Na classificação com um método clássico, utilizei aprendizado de máquina baseado em extração manual de características para classificar os pistaches. O processamento foi realizado no ambiente do Kaggle, onde carreguei os dados pré-processados pelo programa alinha, tanto na versão apenas reduzida quanto na versão alinhada verticalmente.

Para representar as imagens, empreguei o descritor HOG (Histogram of Oriented Gradients), que extrai informações estruturais com base na distribuição de gradientes locais. Converti cada imagem para tons de cinza e calculei os vetores HOG com 9 orientações, células de 8×8 pixels, blocos de 2×2 e normalização L2-Hys. O HOG foi escolhido por ser eficiente, robusto a variações de iluminação e eficaz na representação de contornos, o que o torna adequado para distinguir os formatos dos dois tipos de pistache, por exemplo.

Com os vetores de características gerados, associei rótulos binários (0 para Kirmizi, 1 para Siirt) e dividi o conjunto de dados em 200 imagens para treino e 1948 para teste, respeitando a proporção estipulada e mantendo o balanceamento entre as classes por estratificação. Para classificar os dados, utilizei um classificador SVM (Support Vector Machine) com kernel RBF. Essa escolha se deu pela capacidade do SVM de lidar bem com conjuntos de dados pequenos e sua eficácia ao separar dados em espaços de alta dimensão, como é o caso dos vetores HOG.

Após treinar o SVM com as imagens reduzidas, repeti exatamente o mesmo processo com as imagens alinhadas verticalmente. Em ambos os casos, utilizei validação cruzada com 10 partições para obter uma estimativa mais confiável da acurácia média do modelo.

Ao final do teste, obtive uma acurácia média de 89,20 % com as imagens reduzidas e de 89,52 % com as alinhadas verticalmente, demonstrando que o alinhamento não resultou em uma melhoria significativa no desempenho do SVM para a classificação das imagens.

4 Classificação com Rede Neural Convolucional (Transfer Learning)

Link para notebook no Kaggle (reduzidas)
Link para notebook no Kaggle (alinhadas)

Diante da quantidade reduzida de dados de treinamento (apenas 200 imagens), a construção de uma rede neural convolucional do zero resultou em desempenho significativamente inferior ao do classificador SVM. Isso ocorre porque redes profundas exigem grandes volumes de dados para aprender representações robustas e generalizáveis. Para contornar esse desafio, optei por aplicar transfer learning com fine-tuning com a arquitetura InceptionV3, que já foi previamente treinada em um grande conjunto de imagens (ImageNet) e é mais leve que outras redes populares como a VGG16, o que ajuda a evitar o overfitting em conjuntos pequenos como este.

Comecei carregando as imagens reduzidas (64×64 pixels) e as alinhadas e convertendo-as para valores entre 0 e 1. Em seguida, redimensionei as imagens para 150×150 pixels para atender ao requisito mínimo da entrada da InceptionV3. Também converti o canal de cores de BGR para RGB, já que o modelo pré-treinado foi ajustado com essa convenção de cor.

Para construir a rede, carreguei a InceptionV3 sem as camadas de classificação originais (include_top=False) e adicionei no topo um novo classificador personalizado composto por uma camada de pooling global (GlobalAveragePooling2D), seguida de uma camada densa com 512 unidades e função de ativação ReLU para regularização. Finalizei com uma camada densa com duas saídas e ativação softmax, representando as classes Kirmizi e Siirt.

Congelei as primeiras 249 camadas da InceptionV3, mantendo apenas as últimas camadas treináveis, o que permitiu adaptar as representações finais da rede ao conjunto específico de pistaches sem perder o conhecimento visual aprendido no ImageNet. Compilei o modelo com o otimizador Adam e uma taxa de aprendizado de 0.0001, utilizando a perda categorical_crossentropy e a métrica de acurácia.

Para melhorar o desempenho e reduzir o risco de overfitting, apliquei técnicas de data augmentation usando o ImageDataGenerator do Keras. Gerei variações nas imagens de treino com rotações, deslocamentos, zooms e reflexões horizontais. Além disso, utilizei dois callbacks: EarlyStopping, para interromper o treino caso a acurácia de validação parasse de melhorar, e ReduceLROnPlateau, para reduzir a taxa de aprendizado automaticamente ao detectar estagnação.

Treinei o modelo por até 50 épocas, com batch size de 16, mas interrompi o processo automaticamente com EarlyStopping ao atingir o melhor desempenho em validação. Após o treinamento, avaliei o modelo no conjunto de teste (1.948 imagens) e obtive uma acurácia final média de 88% tanto para o conjunto reduzido

quanto para o alinhado. Também gerei o relatório de classificação com precisão, recall e F1-score para cada classe, além de exibir a matriz de confusão para análise qualitativa do desempenho.

Algo curioso que observei é que o modelo apresentou melhor desempenho na classificação da classe Kirmizi quando utilizei imagens apenas reduzidas, enquanto obteve resultados superiores na classificação da classe Siirt com imagens alinhadas. Essa diferença pode ser explicada pela forma como a CNN lida com a variação espacial e estrutural presente nas imagens. Como redes convolucionais extraem padrões visuais por meio de filtros treináveis aplicados em diferentes regiões da imagem, o alinhamento vertical das imagens reduz a variabilidade posicional e angular dos objetos. Isso beneficia especialmente a classe Siirt, que provavelmente apresentava maior variação na orientação original das imagens, dificultando a extração de padrões consistentes. Já os pistaches Kirmizi parecem ter sido fotografados de maneira mais uniforme no dataset, com uma orientação natural já predominante. Nesse caso, o alinhamento acabou removendo pistas visuais úteis que a rede aprendeu na versão original das imagens, o que justifica a queda no desempenho para essa classe. Portanto, o alinhamento se mostrou vantajoso para classes mais heterogêneas e prejudicial quando já existia uma consistência estrutural na base original.

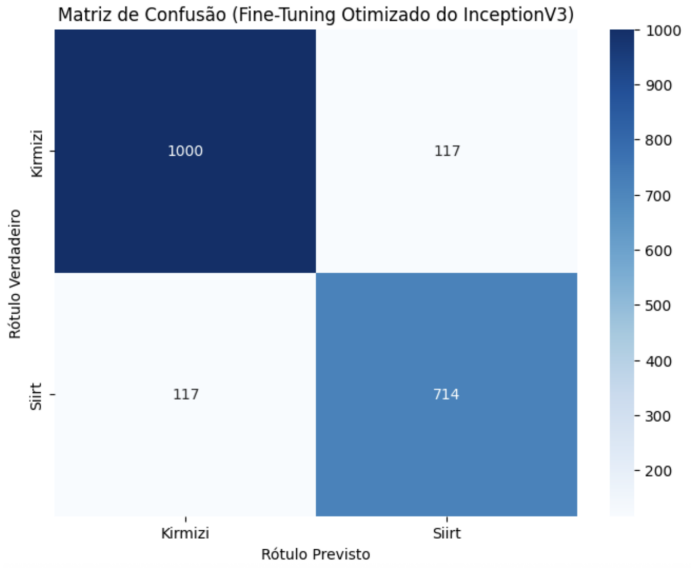

Figura 2: Matriz de confusão com imagens reduzidas

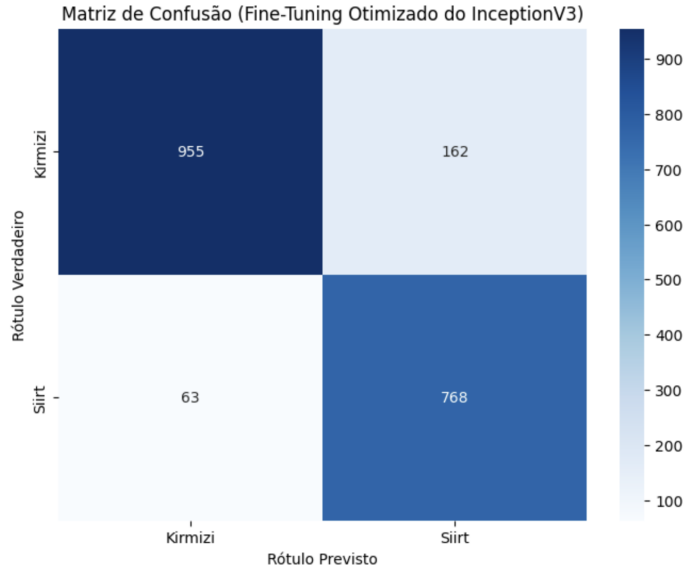

Figura 3: Matriz de confusão com imagens alinhadas

5 Conclusão

Neste trabalho, comparei duas abordagens para a classificação de pistaches (Kirmizi e Siirt): um método clássico com HOG e SVM, e outro baseado em redes convolucionais utilizando transfer learning e fine-tuning com a InceptionV3. O pré-processamento, realizado no programa Alinha, foi essencial para padronizar as imagens por meio do redimensionamento e do alinhamento vertical.

O método clássico apresentou resultados satisfatórios, mesmo com um conjunto reduzido de dados de treino, e demonstrou leve melhoria com as imagens alinhadas. Por outro lado, a abordagem convolucional, inicialmente testada com uma arquitetura própria simples, teve desempenho insatisfatório, o que motivou a adoção de transfer learning. Com a utilização da InceptionV3 pré-treinada, os resultados foram significativamente superiores e mais consistentes, mesmo mantendo o conjunto de dados pequeno.

Essa estratégia baseada em transfer learning mostrou-se altamente eficaz, superando amplamente o desempenho das redes convolucionais construídas do zero. A InceptionV3 destacou-se por sua arquitetura otimizada, com profundidade balanceada e bom uso de parâmetros, favorecendo a generalização dentro do domínio visual específico das imagens de pistache.

Ao final, conclui que o alinhamento vertical e a remoção das bordas pretas — embora úteis para padronização visual — não influenciaram o desempenho dos modelos tanto quanto se esperava. No caso do modelo clássico, houve um ganho modesto; já para a InceptionV3, as imagens alinhadas tiveram um desempenho melhor para os pistaches Siirt e as reduzidas para os pistaches Kirmizi.

Taxa de Erros	**Reduzida**	**Alinhada**
Clássico	89,20%	89,52%
Convolucional	87,80%	88,10%

Tabela 1: Taxa de Erros Obtidas