HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

_____ o0o _____

BÀI TẬP LỚN NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Tên đề tài: Thuật toán K-Nearest Neighbors cho bài toán phân loại màu ảnh

LỚP : N10
Số thứ tự nhóm: 03

Trần Thị Thu Phương	**MSSV: B21DCAT151**
Mầu Nhân Tú	**MSSV: B21DCCN746**
Trần Việt Hà	**MSSV: B21DCAT079**
Nguyễn Hồng Đăng	**MSSV: B21DCAT051**
Đỗ Thị Quỳnh	**MSSV: B21DCCN644**

Giảng viên hướng dẫn: Ths. Hoài Thư

HÀ NỘI, 05/2024

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành nhất đến tất cả những người đã hỗ trợ và giúp đỡ chúng em hoàn thành bài tập lớn này.

Trước hết, chúng em xin bày tỏ lòng biết ơn sâu sắc đến cô Vũ Hoài Thư, người đã tận tình hướng dẫn, cung cấp những kiến thức quý báu cũng như những góp ý, nhận xét kịp thời để chúng em có thể hoàn thiện bài tập này. Kiến thức và kinh nghiệm mà cô đã truyền đạt đã giúp chúng em không chỉ nâng cao về mặt kiến thức mà còn về các kĩ năng chuyên môn khác.

Bên cạnh đó, chúng em muốn gửi lời cảm ơn đến chính các bạn trong nhóm, những người đã trực tiếp đóng góp thời gian, công sức cũng như cùng nhau chia sẻ và thảo luận trong suốt quá trình làm bài. Mỗi thành viên đã đóng góp những kỹ năng, kiến thức và sự nỗ lực để đảm bảo rằng bài tập này được hoàn thành một cách tốt nhất.

Một lần nữa, xin chân thành cảm ơn tất cả mọi người!

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	**Ý nghĩa**
BTL	Bài tập lớn
KNNs	Thuật toán k-Nearest Neighbors
RGB	Kênh màu Red, Green, Blue

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Với lượng lớn hình ảnh chụp ngày càng tăng, phân loại màu ảnh là một bài toán quan trọng trong nhiều lĩnh vực khác nhau đặc biệt trong các ứng dụng xử lý ảnh.

Việc phân loại ảnh theo màu sắc giúp sắp xếp, quản lý và tìm kiếm ảnh nhanh chóng hơn, hiệu quả hơn. Như chúng ta đã biết, phân loại màu ảnh là bước đầu tiên cho nhiều ứng dụng xử lý ảnh khác nhau như: cân bằng màu sắc, tăng cường màu sắc, loại bỏ nhiễu màu, chuyển đổi màu sắc. Không những vậy, phân loại màu ảnh còn được sử dụng để trích xuất thông tin từ hình ảnh, chẳng hạn như nhận diện đối tượng, phân tích nội dung hình ảnh, kiểm tra chất lượng ảnh.

Ngoài xử lý ảnh, bài toán phân loại màu ảnh có thể được áp dụng trong nhiều lĩnh vực khác nhau như:

- **Y tế**: phân loại ảnh chụp X-quang, MRI để hỗ trợ chuẩn đoán bệnh
- **Nông nghiệp**: phân loại cây trồng, theo dõi sức khỏe cây
- **Khảo cổ học**: phân tích và phân loại các di vật khảo cổ

Việc giải quyết bài toán này giúp mang lại nhiều lợi ích cho con người:

- **Tự động hóa các tác vụ thủ công**: giúp giảm bớt gánh nặng cho con người trong việc phân loại ảnh thủ công, tiết kiệm thời gian và công sức
- **Nâng cao hiệu quả**: cải thiện độ chính xác và hiệu quả của việc phân loại ảnh đặc biệt khi xử lý với số lượng ảnh quá lớn

Như vậy, phân loại màu ảnh là một bài toán quan trọng với nhiều lĩnh vực trong đời sống. Việc giải quyết bài toán này mang lại nhiều lợi ích cho con người và có tiềm năng mở ra nhiều ứng dụng mới trong tương lai.

1.2 Mục tiêu và phạm vi đề tài

Hiện nay, các sản phẩm nông nghiệp như rau, trái cây đang được sử dụng rộng rãi trên toàn thế giới và như cầu sử dụng cũng ngày càng tăng. Đi đôi với đó là yêu cầu về các sản phẩm ngày càng cao hơn như không có thuốc trừ sâu, quả phải có vị ngọt,... Thế nên các ứng dụng công nghệ hỗ trợ cho việc nuôi trồng thực phẩm cũng càng cấp thiết hơn bao giờ. Rất nhiều trường đại học trên khắp thế giới đã thực hiện các nghiên cứu về việc ứng dụng công nghệ vào việc phân tích màu sắc hỗ trợ cho trồng trọt và chăm sóc cây trồng:

Phân tích tình trạng cây trồng: Một hệ thống phân tích màu sắc lá cây trong

các trang trại lớn tại Ấn Độ đã giúp giảm 20% lượng thuốc trừ sâu nhờ vào việc phát hiện sớm các dấu hiệu bệnh tật. Ứng dụng này không chỉ giúp nông dân phát hiện sớm bệnh tật mà còn tối ưu hóa việc sử dụng phân bón và thuốc trừ sâu, từ đó tăng hiệu quả sản xuất và giảm chi phí.

Phân loại nấm bệnh trên cây trồng: Nghiên cứu tại Đại học Wageningen (Hà Lan) đã sử dụng hệ thống phân loại màu sắc để phân loại các loại nấm bệnh dựa trên màu sắc và hình dạng của chúng trên lá cây. Hệ thống này đạt độ chính xác 90%, giúp nông dân nhận biết và xử lý nấm bệnh kịp thời, từ đó bảo vệ mùa màng và tăng năng suất.

Phân tích màu sắc đất: Một nghiên cứu tại Đại học Purdue đã áp dụng việc phân loại màu ảnh để phân tích màu sắc đất từ hình ảnh vệ tinh, giúp xác định các vùng đất bị suy thoái hoặc cần cải tạo. Hệ thống này đã giúp giảm 15% chi phí cải tạo đất nhờ vào việc xác định chính xác các khu vực cần can thiệp.

Phân tích độ chín của cây trồng: Một nghiên cứu tại Đại học Tokyo đã ứng dụng hệ thống xử lý ảnh và phân loại màu để phân tích màu sắc của cây cà phê nhằm xác định độ chín. Hệ thống này giúp nông dân xác định thời điểm thu hoạch chính xác, tăng năng suất và chất lượng hạt cà phê.

Qua các nghiên cứu trên ta có thể thấy được các hiệu quả thực tế của việc ứng dụng công nghệ phân loại màu sắc vào trong nông nghiệp trồng trọt tuy nhiên vẫn còn một số hạn chế:

Độ chính xác của thuật toán có thể không cao trong các bối cảnh phức tạp, đặc biệt khi màu sắc của các đối tượng phân loại tương đồng nhau.

Việc xử lý dữ liệu chậm và tiêu tốn nhiều tài nguyên khi xử lý các tập dữ liệu lớn, do phải tính toán khoảng cách cho tất cả các điểm dữ liệu.

1.3 Định hướng giải pháp

Nhóm chúng em xin đưa ra một số hướng giải quyết các vấn đề trên có thể khả thi:

Phương pháp và thuật toán: Chúng em đề xuất sử dụng thuật toán K-Nearest Neighbors (KNN) để phân loại màu sắc trong ảnh, với mục tiêu áp dụng vào lĩnh vực nông nghiệp trong tương lai.

Tăng độ chính xác: Áp dụng các kỹ thuật tiền xử lý dữ liệu như giảm nhiễu và tăng cường dữ liệu, kết hợp KNN với các thuật toán học sâu để tạo ra hệ thống phân loại lai nhằm cải thiện độ chính xác.

Tối ưu hóa hiệu suất: Sử dụng các cấu trúc dữ liệu như KD-tree hoặc Ball tree

để giảm thời gian tính toán khoảng cách, cải thiện hiệu suất của KNN khi xử lý các tập dữ liệu lớn

Giải pháp: Chúng em sẽ phát triển một hệ thống phân loại màu sắc đơn giản dựa trên thuật toán KNN. Hiện tại, chúng em tập trung vào việc phân loại màu sắc trong ảnh, nhưng trong tương lai, chúng em dự định mở rộng ứng dụng của hệ thống này vào lĩnh vực nông nghiệp.

Đóng góp chính: Bằng cách phát triển một hệ thống phân loại màu sắc đơn giản, chúng tôi tạo ra một cơ sở để mở rộng ứng dụng vào lĩnh vực nông nghiệp trong tương lai. Hệ thống này có thể hỗ trợ nông dân trong việc phát hiện sớm bệnh tật trên cây trồng, tối ưu hóa việc quản lý và sản xuất nông nghiệp

Kết quả đạt được: Mặc dù hiện tại chỉ có khả năng phân loại màu sắc trong ảnh, nhưng hệ thống này có tiềm năng để phát triển và mở rộng ứng dụng vào lĩnh vực nông nghiệp. Trong tương lai, chúng tôi hy vọng rằng hệ thống này sẽ đóng góp vào việc cải thiện sản xuất nông nghiệp và giảm chi phí cho nông dân.

1.4 Bố cục bài tập lớn

Phần còn lại của báo cáo bài tập lớn này được tổ chức như sau.

Chương 2 giới thiệu tổng quan về bài toán phân loại màu ảnh, các khái niệm cơ bản và lý thuyết nền tảng về thuật toán KNNs và ứng dụng của thuật toán KNNs trong bài toán phân loại màu ảnh.

Trong Chương 3, nhóm em sẽ trình bày chi tiết cách xây dựng, đánh giá mô hình Machine Learning K-Nearest Neighbors, minh họa cách áp dụng mô hình này vào bài toán phân loại màu ảnh một cách hiệu quả.

Chương 4 sẽ nêu lên ưu và nhược điểm của việc ứng dụng thuật toán K-Nearest Neighbors(KNNs) so với các phương pháp hiện đại như Mạng Nơron Tích Chập (CNN) trong các lĩnh vực của khoa học máy tính và trí tuệ nhân tạo. Đồng thời chúng em cũng đề cập tới định hướng phát triển tối ưu hóa hiệu suất KNN, mở rộng ứng dụng vào các lĩnh vực như nhận diện màu sắc và phân loại văn bản, và xây dựng công cụ hỗ trợ để áp dụng kiến thức vào thực tiễn trong tương lai.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Chương 1 đã cung cấp cái nhìn tổng quan về BTL, bao gồm lý do chọn đề tài, mục tiêu, phạm vi nghiên cứu và phương pháp tiếp cận. Để hiểu rõ hơn và có cơ sở vững chắc cho nghiên cứu, Chương 2 sẽ trình bày các khái niệm cơ bản và lý thuyết nền tảng liên quan. Những cơ sở lý thuyết này sẽ giúp chúng ta phân tích và thảo luận vấn đề một cách chi tiết và rõ ràng.

2.1 Giới thiệu về bài toán phân loại màu ảnh

Phân loại màu ảnh là gì:

Phân loại màu ảnh là một nhiệm vụ quan trọng trong xử lý ảnh số, có ứng dụng rộng rãi trong nhiều lĩnh vực như y tế, công nghiệp, nông nghiệp, v.v. Mục tiêu của bài toán là tự động phân loại các pixel trong ảnh thành các lớp màu khác nhau, ví dụ như màu đỏ, xanh lá cây, xanh lam, v.v.

Các phương pháp:

Có nhiều phương pháp khác nhau để giải quyết bài toán phân loại màu ảnh, bao gồm:

- Phương pháp dựa trên ngưỡng: Phương pháp này sử dụng một ngưỡng giá trị để phân chia các pixel thành các lớp màu khác nhau. Ví dụ, tất cả các pixel có giá trị màu đỏ lớn hơn một ngưỡng nhất định có thể được phân loại là màu đỏ.
- Phương pháp phân cụm: Phương pháp này sử dụng các thuật toán phân cụm để nhóm các pixel có màu sắc tương tự nhau thành các lớp màu khác nhau.
- Phương pháp học máy: Phương pháp này sử dụng các mô hình học máy để học cách phân loại các pixel thành các lớp màu khác nhau từ một tập dữ liệu huấn luyện.

Ứng dụng:

Phân loại màu ảnh có nhiều ứng dụng trong thực tế, bao gồm:

- Phân tích hình ảnh y tế: Phân loại màu ảnh có thể được sử dụng để phân tích hình ảnh y tế, ví dụ như hình ảnh X-quang, để phát hiện các bệnh lý.
- Kiểm soát chất lượng trong công nghiệp: Phân loại màu ảnh có thể được sử dụng để kiểm soát chất lượng sản phẩm trong công nghiệp, ví dụ như để phát hiện các lỗi trong sản phẩm..
- Phân loại nông sản: Phân loại màu ảnh có thể được sử dụng để phân loại nông sản, ví dụ như để phân loại trái cây theo độ chín.

Thử thách:

Bài toán phân loại màu ảnh có thể gặp một số thử thách, bao gồm:

- Ánh sáng: Ánh sáng có thể ảnh hưởng đến màu sắc của các pixel trong ảnh, dẫn đến kết quả phân loại không chính xác.
- Độ nhiễu: Độ nhiễu trong ảnh có thể ảnh hưởng đến kết quả phân loại.
- Sự phức tạp của cảnh: Các cảnh phức tạp với nhiều đối tượng và màu sắc khác nhau có thể khó phân loại hơn.

Nghiên cứu:

Nghiên cứu về bài toán phân loại màu ảnh vẫn đang được tiếp tục để cải thiện độ chính xác và hiệu quả của các phương pháp phân loại. Một số hướng nghiên cứu hiện nay bao gồm:

- Sử dụng dữ liệu lớn: Dữ liệu lớn có thể được sử dụng để huấn luyện các mô hình học máy tốt hơn cho bài toán phân loại màu ảnh.
- Kết hợp nhiều phương pháp: Kết hợp nhiều phương pháp phân loại khác nhau có thể giúp cải thiện độ chính xác và hiệu quả của việc phân loại màu ảnh.

2.2 Thuật toán K-Nearest Neighbors

2.2.1 Nguyên tắc chung

- Một số tên gọi khác của thuật toán
  + Instance-based learning
  + Lazy learning
  + Memory-based learning
- Gồm một số kỹ thuật học khác nhau:
  + Thuật toán k-láng giềng gần nhất (k-NN)
  + Suy diễn theo trường hợp (case-based reasoning)

2.2.2 Phương pháp KNN

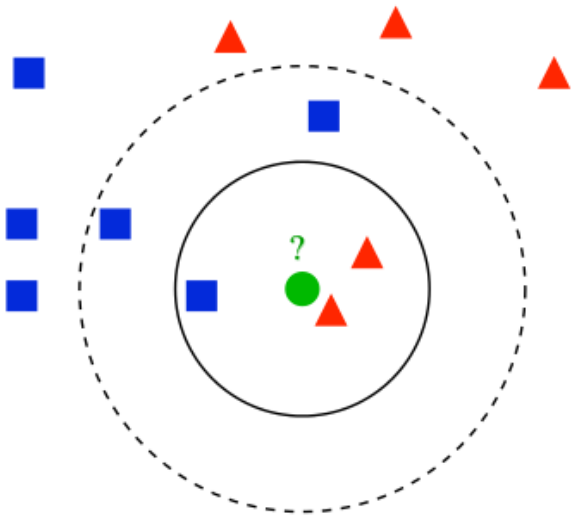
K-láng giềng gần nhất (k-NN) là phương pháp tiêu biểu của học dựa trên ví dụ.

- Ý tưởng:
  - Với một tập ví dụ học
    + Đơn giản là lưu lại các ví dụ học
    + Chưa xây dựng một mô hình rõ ràng và tổng quát của hàm mục tiêu cần học

- Đối với một ví dụ cần phân loại/dự đoán: Xét quan hệ giữa ví dụ đó với các ví dụ học để gán giá trị của hàm mục tiêu.

- Biểu diễn đầu vào của bài toán
  - Mỗi ví dụ x được biểu diễn là một vector n chiều
  - $x = (x_1, x_2, \ldots, x_n)$ trong đó $x_i \in \mathbb{R}$ là một số thực.
- Có thể áp dụng với các bài toán phân loại hoặc hồi quy

1. **Phương pháp KNN trong bài toán phân loại**
   - Giai đoạn học (huấn luyện): Lưu lại các ví dụ trong tập dữ liệu
   - Giai đoạn phân lớp: Để phân lớp cho ví dụ z:
     + Với mỗi ví dụ x thuộc tập dữ liệu huấn luyện, tính khoảng cách giữa x và z
     + Xác định tập láng giềng gần nhất của z
     + Phân z vào lớp chiếm số đông trong số các lớp của các ví dụ học



Hình 2.1: Kết quả phân loại của k-NN

Kết quả phân loại của thuật toán k-NN được minh họa trên hình 2.1 cho trường hợp phân loại 2 lớp.

- Với $k = 3$, ví dụ được phân loại thành tam giác
- Với $k = 5$, ví dụ được phân loại thành hình vuông

2. **Phương pháp KNN trong bài toán hồi quy**
   - Giai đoạn học (huấn luyện): Lưu lại các ví dụ trong tập dữ liệu huấn luyện
   - Giai đoạn dự đoán: Để dự đoán giá trị đầu ra cho ví dụ mới z:

+ Với mỗi ví dụ x thuộc tập dữ liệu huấn luyện, tính khoảng cách giữa x và z

+ Xác định tập láng giềng gần nhất của z là NB(z)

+ Dự đoán giá trị đầu ra đối với z:

$$y_z = \frac{1}{k} \sum_{x \in NB(z)} y_x$$

3. Một số hàm tính khoảng cách

- Hàm Euclid

$$d(x, z) = \sqrt{\sum_{i=1}^{n} (x_i - z_i)^2}$$

- Hàm Manhattan

$$d(x, z) = \sum_{i=1}^{n} |x_i - z_i|$$

2.2.3 Một số lưu ý với thuật toán KNN

Chọn giá trị của K

Lựa chọn giá trị của K là một yếu tố quan trọng. Giá trị của K sẽ ảnh hưởng đến hiệu suất của mô hình. K quá nhỏ có thể dẫn đến overfitting, trong khi K quá lớn có thể dẫn đến underfitting. Cần thực hiện kiểm định chéo để chọn giá trị K tốt nhất cho bài toán cụ thể

Chọn phương pháp đo khoảng cách

Khoảng cách giữa các điểm dữ liệu có thể được đo bằng nhiều phương pháp khác nhau như khoảng cách Euclidean, khoảng cách Mahalanobis, hoặc khoảng cách cosine. Chọn phương pháp đo khoảng cách phù hợp với đặc điểm của dữ liệu và bài toán cụ thể.

Xử lý dữ liệu

KNN cần dữ liệu được chuẩn hóa để các đặc trưng có cùng thang đo. Việc này đặc biệt quan trọng đối với các đặc trưng có đơn vị đo khác nhau. Loại bỏ các giá trị nhiễu hoặc xử lý dữ liệu bị thiếu để tránh ảnh hưởng đến kết quả dự đoán.

Tính toán hiệu suất và độ chính xác

Đánh giá hiệu suất của mô hình KNN bằng cách sử dụng các độ đo phù hợp như độ chính xác, độ phân loại chính xác, độ phủ, hoặc ma trận nhầm lẫn. Cần xem xét cả độ chính xác trên tập huấn luyện và tập kiểm tra để đảm bảo mô hình không bị overfitting

Xử lý số lượng chiều lớn

Khi số chiều của dữ liệu tăng, hiệu suất của KNN có thể giảm do hiệu ứng "curse of dimensionality". Cần xem xét việc sử dụng kỹ thuật giảm chiều dữ liệu như PCA để giảm số chiều của dữ liệu mà vẫn giữ được thông tin quan trọng

Tính toán hiệu suất tính toán

KNN có thể tốn nhiều thời gian tính toán đặc biệt khi số lượng điểm dữ liệu lớn hoặc số chiều của dữ liệu cao. Cần xem xét các phương pháp tối ưu hoá hoặc kỹ thuật giảm chiều dữ liệu để cải thiện hiệu suất tính toán của mô hình

2.2.4 Ưu, nhược điểm của thuật toán KNN

Ưu điểm

- Độ phức tạp tính toán của quá trình training là bằng 0.
- Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
- Không cần giả sử gì về phân phối của các class.

Nhược điểm

- KNN có thể bị ảnh hưởng nhiều bởi các nhiễu hoặc dữ liệu không chuẩn. Các điểm nhiễu có thể ảnh hưởng đến quá trình phân loại, đặc biệt khi giá trị của K nhỏ.
- KNN yêu cầu lưu trữ toàn bộ tập dữ liệu huấn luyện trong bộ nhớ, điều này có thể là một vấn đề với các tập dữ liệu lớn
- KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test. Trong đó việc tính khoảng cách tới từng điểm dữ liệu trong training set sẽ tốn rất nhiều thời gian, đặc biệt là với các cơ sở dữ liệu có số chiều lớn và có nhiều điểm dữ liệu
- Ngoài ra, việc lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN

2.3 Ứng dụng thuật toán KNN cho bài toán phân loại màu ảnh

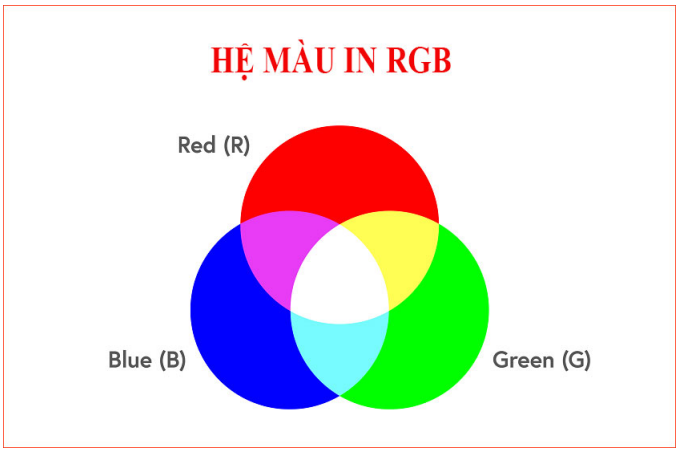
Như đã nêu ở trên, thuật toán KNN là một thuật toán học máy thuộc nhóm giám sát, được sử dụng phổ biến trong các bài toán phân loại và hồi quy. Khi áp dụng KNN vào bài toán phân loại màu ảnh, chúng ta có thể thực hiện theo các bước sau:

Bước 1: Chuẩn bị dữ liệu

– Thu thập dữ liệu ảnh: Chọn một tập hợp các hình ảnh để sử dụng cho việc huấn luyện và kiểm tra.

– Tiền xử lý dữ liệu: Chuyển đổi các hình ảnh sang không gian màu mà bạn muốn làm việc (ví dụ: RGB trong hình 2.2, HSV, Lab, v.v.). Sau đó, chia các

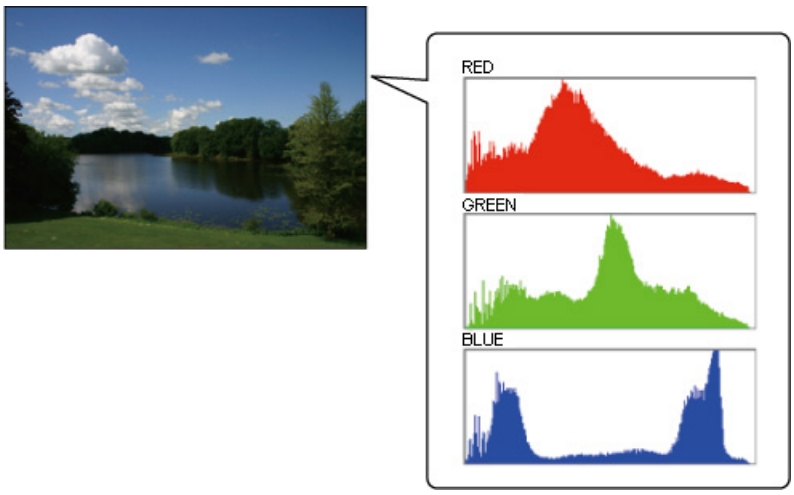
hình ảnh này thành các pixel màu riêng lẻ.



Hình 2.2: Kênh màu RGB

Bước 2: Đặc trưng hóa dữ liệu

– Trích xuất đặc trưng màu: Đối với mỗi pixel màu, lấy giá trị màu làm đặc trưng. Ví dụ: với không gian màu RGB, mỗi pixel sẽ có một vector đặc trưng gồm ba giá trị [R, G, B].



Hình 2.3: Đặc trưng của 1 bức ảnh được lấy theo kênh màu RGB

– Gắn nhãn dữ liệu: Nếu có một tập dữ liệu huấn luyện đã được gắn nhãn, mỗi pixel sẽ có nhãn tương ứng chỉ định màu sắc cụ thể của nó.

	red	green	blue	label
0	20	139	240	Blue
1	174	83	72	Brown
2	144	249	131	Green
3	168	25	156	Pink
4	30	182	136	Green
5	199	150	175	Pink
6	199	93	154	Pink
7	231	243	25	Yellow
8	48	213	76	Green
9	38	3	64	Blue
10	82	181	56	Green
11	249	99	108	Pink

Hình 2.4: Gán nhãn cho các mẫu

Bước 3: Áp dụng thuật toán KNN

– Chọn giá trị K: Quyết định số lượng hàng xóm gần nhất sẽ được xem xét. Giá trị K có thể được lựa chọn dựa trên thử nghiệm và đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra.

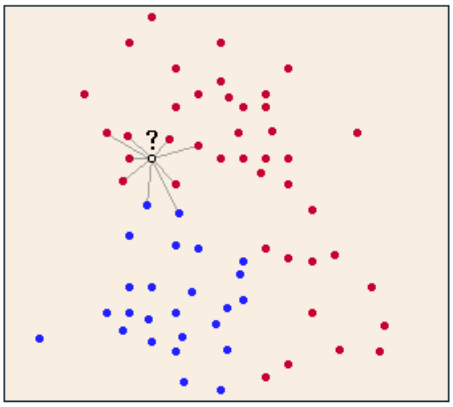
– Xây dựng mô hình KNN: Sử dụng tập dữ liệu huấn luyện để xây dựng mô hình KNN. Đây là quá trình mà mô hình ghi nhớ tất cả các điểm dữ liệu huấn luyện và nhãn của chúng.

Bước 4: Phân loại màu sắc

– Phân loại điểm mới: Khi có một điểm ảnh mới cần phân loại, tính khoảng cách giữa điểm đó với tất cả các điểm trong tập dữ liệu huấn luyện (thường sử dụng khoảng cách Euclid).

– Tìm K hàng xóm gần nhất: Chọn ra K điểm ảnh gần nhất.

– Bỏ phiếu cho nhãn: Nhãn của điểm mới được xác định dựa trên nhãn chiếm đa số của k láng giềng gần nhất kia



Hình 2.5: Dự đoán sử dụng KNN

Bước 5: Đánh giá và hiệu chỉnh mô hình

– Đánh giá mô hình: Sử dụng các tiêu chí đánh giá như độ chính xác, độ nhạy, độ đặc hiệu, v.v., để đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra.

– Hiệu chỉnh mô hình: Nếu cần, điều chỉnh các tham số của mô hình như giá trị K hoặc cách thức tính khoảng cách để cải thiện hiệu suất.

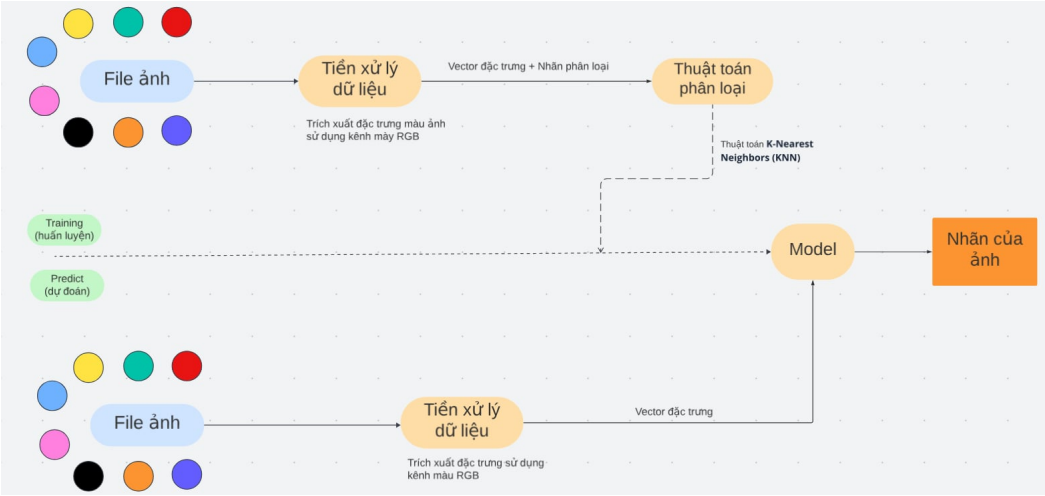
CHƯƠNG 3. THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

Trong Chương 2, nhóm em đã giới thiệu thuật toán KNN và ứng dụng của nó trong việc phân loại màu ảnh. Ở Chương 3, nhóm em sẽ đi sâu vào chi tiết cách xây dựng và đánh giá mô hình, đồng thời minh họa cách áp dụng mô hình này vào bài toán phân loại màu ảnh một cách hiệu quả.

3.1 Sơ đồ phương pháp phân loại

Trong bài tập lớn này, màu sắc được phân loại bằng cách sử dụng thuật toán phân loại Machine Learning K-Nearest Neighbors. Mô hình phân loại này được huấn luyện bởi các giá trị biểu đồ màu R, G, B của hình ảnh. Quá trình thực hiện được mô tả bằng hình dưới đây

Cụ thể:



Hình 3.1: Sơ đồ phân loại của mô hình

Trong sơ đồ hình 3.1:

- Giai đoạn tiền xử lý dữ liệu: đây là giai đoạn đặc trưng hóa dữ liệu, hay biểu diễn mẫu ví dụ x thành n các đặc trưng của nó để đem đi phân loại
- Giai đoạn phân loại: sử dụng thuật toán KNN

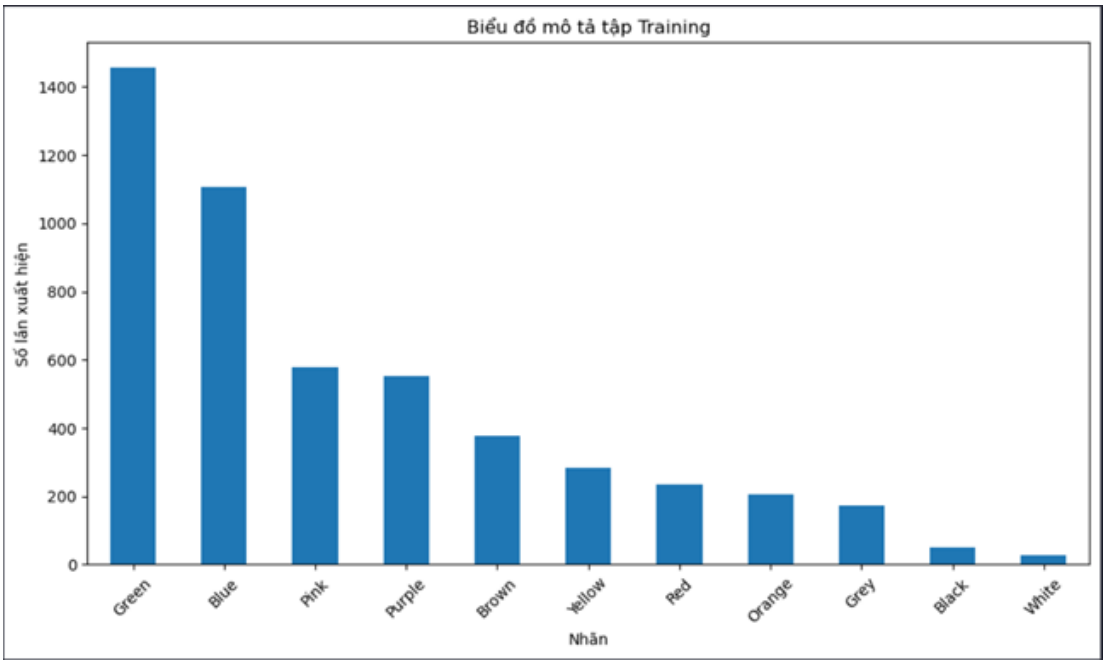
3.2 Mô tả tập dữ liêu

Tập dữ liệu của mô hình bao gồm 2 loại: tập Training, tập Test.

Tập Training là file csv chứa 3 đặc trưng được phân tích theo biểu đồ màu RGB và nhãn của 5053 màu được chia làm 11 nhóm màu: đen, xanh dương, xanh lá, nâu, xám, cam, hồng, tím, đỏ, trắng và vàng với số lượng được mô tả trong hình 3.2. Mỗi màu là một dòng trong file csv, cụ thể trong hình 3.3.

Tập Test gồm 307 file ảnh với đầy đủ 11 nhóm màu kể trên, chưa được trích xuất đặc trưng.

Tập Training được sử dụng để huấn luyện mô hình còn tập Test được sử dụng để đánh giá độ chính xác sau khi huấn luyện.



Hình 3.2: Biểu đồ mô tả tập dữ liệu training

	red	green	blue	label
0	20	139	240	Blue
1	174	83	72	Brown
2	144	249	131	Green
3	168	25	156	Pink
4	30	182	136	Green
5	199	150	175	Pink
6	199	93	154	Pink
7	231	243	25	Yellow
8	48	213	76	Green

Hình 3.3: Dữ liệu Training

3.3 Đặc trưng hóa dữ liệu

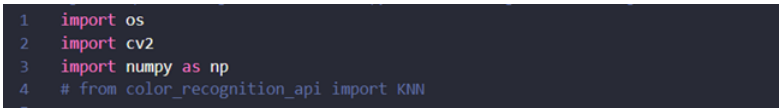
Như đã nêu ở phần trước, mô hình này sử dụng biểu đồ màu RGB để trích xuất các đặc trưng màu sắc của ảnh.

3.3.1 Khai báo các thư viện

Thư viện os: được sử dụng trong quá trình duyệt từng tệp trong thư mục.

Thư viện cv2: được sử dụng trong quá trình xử lý ảnh

Thư viện numpy: được sử dụng để tính toán số học và ma trận



Hình 3.4: Khai báo các thư viện cần thiết khi trích xuất đặc trưng màu ảnh

3.3.2 Xây dựng hàm

a, Đối với mẫu dữ liệu Training/Test

Để lấy các đặc trưng về màu sắc của ảnh, ta sử hàm color_histogram_of_imag để tính toán các histogram của 1 ảnh tên img_nam theo biểu đồ màu RGB, sau đó đặc trưng của mẫu dữ liệu sẽ được ghi vào tệp testing.data cùng với nhãn mô tả.

Để đọc ảnh từ máy, tách ảnh thành các kênh màu B, G, R và tính toán các histogram của các kênh màu, ta sử dụng các hàm trong thư viện OpenCV. Nhãn của mẫu dữ liệu được gán theo tên thư mục đang chứa mẫu đó. Phần cài đặt chi tiết trong hình 3.5

```python
def color_histogram_of_image(img_name):
    '''
    Hàm này để tính toán các histogram của 1 ảnh tên img_name theo RGB
    và sau đó được ghi vào tệp traning.data cùng với nhãn mô tả của nó
    '''

    # gán nhãn cho các file: nếu tên ảnh/đường dẫn chứa màu nào thì sẽ gán nhãn của ảnh đó là màu ý
    label =''
    for file_name in os.listdir('./testing_image'):
        if(file_name in img_name):
            label = file_name
            break
    # load the image
    image = cv2.imread(img_name) #Đọc ảnh từ đường dẫn 'img_name' bằng OpenCV
    chans = cv2.split(image) #Tach anh mau thanh cac kenh mau B, G, R để lấy đặc trưng
    colors = ('b', 'g', 'r')
    features = []
    feature_data = ''
    counter = 0
    for (chan, color) in zip(chans, colors):
        counter = counter + 1
        hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
        elem = np.argmax(hist)
        if counter == 1:
            blue = str(elem)
        elif counter == 2:
            green = str(elem)
        elif counter == 3:
            red = str(elem)
            feature_data = red + ',' + green + ',' + blue
    with open('./testing_dataset/testing.data', 'a') as myfile:
        myfile.write(feature_data + ',' + label + '\n')

```

Hình 3.5: Trích xuất đặc trưng của mẫu dữ liệu Training

Lần lượt duyệt qua từng tệp ảnh và lưu nó vào file testing.data để sử dụng trong quá trình đánh giá mô hình. Cài đặt được mô tả cụ thể như hình 3.6

```python
def feature_test():
    '''
    hàm này để gán nhãn cho tất cả cac ảnh
    đồng thời nó sử dụng hàm color_histogram_of_image(img_name) để trích xuất các histogram
    '''
    for f in os.listdir('./testing_image/Black'):    # black color test images
        color_histogram_of_image('./testing_image/Black/' + f)
    for f in os.listdir('./testing_image/Blue'):    # blue color test  images
        color_histogram_of_image('./testing_image/Blue/' + f)
    for f in os.listdir('./testing_image/Brown'):    # brown color test  images
```

Hình 3.6: Tạo vector đặc trưng và nhãn của tệp ảnh

b, Đối với mẫu dữ liệu Predict

Mẫu dữ liệu Predict khác mẫu dữ liệu trong tập Training/Test ở chỗ nó không có nhãn, bởi vậy nhóm em đề xuất hàm color_histogram_of_test_image với chức năng và cách làm tương tự hàm color_histogram_of_imag, tuy nhiên được bỏ đi phần gán nhãn. Cụ thể trong hình 3.7

```python
def color_histogram_of_test_image(test_src_image):
    '''
    Trích xuất đặc tưng màu RGB để đem đi dự đoán
    '''

    # load the image
    image = test_src_image

    chans = cv2.split(image) #chia ảnh thành các kênh màu, theo cv2 thì là 3 kênh màu theo thứ tự BGR
    colors = ('b', 'g', 'r') #tên các kênh màu

    feature_data = '' #chứa đặc trưng màu của 1 ảnh
    counter = 0 #có vai trò để kiểm tra đang ở kênh màu nào
    for (chan, color) in zip(chans, colors):
        counter = counter + 1

        hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
        #hist = cv2.calcHist(images, channels, mask, hitSize, ranges)

        elem = np.argmax(hist) #trả về chỉ số lớn nhất trong mảng hist

        if counter == 1:
            blue = str(elem)
        elif counter == 2:
            green = str(elem)
        elif counter == 3:
            red = str(elem)
            feature_data = red + ',' + green + ',' + blue
            # print(feature_data)

    # Mo file va ghi de len file data
    with open('./testing_dataset/testing_image.data', 'w') as myfile:
        myfile.write(feature_data)
```

Hình 3.7: Trích xuất đặc trưng của mẫu dữ liệu Predict

3.4 Sử dụng phương pháp K-nearest Neighbors để phân loại màu ảnh

Sau khi đã có đặc trưng của các mẫu dữ liệu, ta đưa chúng vào giải thuật học máy KNN để dạy cho máy học. Phần này sẽ trình bày chi tiết về mã giả của giải thuật này.

3.4.1 Tải dữ liệu đưa vào giải thuật KNN

Đầu tiên, ta cần tải tệp lưu dữ liệu từ một đương dẫn trong máy vào giải thuật

```python
def loadData(path):
    """
    Ham isfile(): Kiem tra lieu mot duong dan cho truoc co ton tai hay khong
    Ham os.access(path, mode): kiem tra kha nang truy cap cua mot file nao do
        + Return True or False
        + Cac Mode:
            R_OK: read permission
            F_OK: Existence
            W_OK: Write permission
    """
    if os.path.isfile(path) and os.access(path, os.R_OK):
        with open(path) as csvfile:
            lines = csv.reader(csvfile)
            data = list(lines) # Chuyen file csv ve dang list 2 chieu

            if 'training' in path:
                data.pop(0) # Xoa header neu la tap training

            for x in range(len(data)):
                for y in range(3):
                    data[x][y] = float(data[x][y])
            return data
```

Hình 3.8: Tải dữ liệu

3.4.2 Tính khoảng cách Ơ-clit

Hàm calculateDistance tính khoảng cách Ơ-clit giữa 2 mẫu dữ liệu, cụ thể là được sử dụng cho mẫu dữ liệu mới và mẫu dữ liệu đã học được trong quá trình training. Do các mẫu dữ liệu có 3 đặc trưng nên biến length sẽ được gán bằng 3. Mã giải chi tiết trong hình 3.9

```python
#Tính toán khoảng cách Eulid giữa 2 mẫu với length là số đặc trưng của 1 mẫu
def calculateDistance(var1, var2):
    '''
    #Tính toán khoảng cách Eulid giữa 2 mẫu với length là số đặc trưng của 1 mẫu
    var1: mẫu 1
    var2: mẫu 2
    length: số lượng đặc trưng của 1 mẫu
    '''
    length= 3 # Do một mau co 3 he mau (RGB)
    distance = 0
    for i in range(length):
        distance += pow((var1[i] - var2[i]), 2)
    return math.sqrt(distance)
```

Hình 3.9: Tính khoảng cách của O-clit

3.4.3 Chọn K láng giềng gần nhất

Hàm k_NearestNeighbors chọn ra k láng giềng gần nhất sử dụng hàm calculate-Distance đã nêu ở phần trên để tính toán khoảng cách giữa mẫu dữ liệu đang xét với các mẫu dữ liệu trong tập training. Đầu vào của hàm là tập các đặc trưng của tập training, đặc trưng của mẫu dữ liệu và số lượng láng giềng k cần xét. Kết quả trả về là k các tuple với giá trị 1 là nhãn của láng giềng và giá trị thứ 2 là khoảng

cách giữa 2 mẫu. Mã giải chi tiết trong hình 3.10

```python
#Lấy K láng giềng gần nhất
def k_NearestNeighbors(training_data, test_instance, k):
    '''
    Tìm k láng giềng gần nhất đối với 1 tập dữ liệu mới
    training_feature_vector: các mẫu đã có trong tập dữ liệu
    testIntance: mẫu mới cần xác định nhãn
    k: số lượng láng giềng
    '''
    list_neighbor =[] # list chua cac tuple voi gia tri dau la label, gia tri 2 la khoang cach
    for x in training_data:
        distance = calculateDistance(x, test_instance) # Lan luot tinh khoang cach giua du lieu
        list_neighbor.append((x[-1], distance))
    list_neighbor.sort(key=lambda x: x[1])
    return list_neighbor[:k] # Tra ve k tuple dau tien
```

Hình 3.10: Chọn K láng giềng gần nhất

3.4.4 Xác định nhãn chiếm đa số

Để xác định nhãn chiếm đa số, ta sử dụng hàm findMostOccur nhận đầu vào là danh sách các tuple được trả về trong hàm k_NearestNeighbors. Đơn giản nhất, ta sử dụng một dictionary frequency với key là nhãn và value là số lần xuất hiện của nhãn đó. Frequency được sắp xếp theo giá trị value giảm dần. Kết quả trả về là nhãn với số lần xuất hiện nhiều nhất. Mã giải chi tiết trong hình 3.11

```python
def findMostOccur(list_neighbor):
    frequency = {} # 1 dict luu tan suat xuat hien
    for neighbor in list_neighbor:
        if neighbor[0] in frequency:
            frequency[neighbor[0]] +=1
        else:
            frequency[neighbor[0]] = 1

    # Sap xep theo tan suat xuat hien
    res = sorted(frequency.items(), key= lambda x: x[1], reverse= True)
    return res[0][0] # Tra ve gia tri dau tien cua dict (label)
```

Hình 3.11: Xác định nhãn chiếm đa số trong k láng giềng

3.4.5 Dự đoán nhãn

Sau khi đã xây dựng các hàm cần thiết, ta áp dụng chúng để dự đoán màu chủ đạo (nhãn) cho 1 file ảnh bất kỳ. Đặc trưng của file ảnh lưu trong tệp có đường dẫn là testing_path. Các đặc trưng này được đưa qua KNN để phân loại, nhãn của ảnh sẽ là nhãn chiếm đa số của k láng giềng gần nhất.

```python
def classify_img(testing_path):
    training_vector=[]
    testing_vector = []

    training_vector = loadData('./training_dataset/data.csv')
    testing_vector = loadData(testing_path)
    # print(testing_vector)
    k = 5
    list_neighbor = k_NearestNeighbors(training_vector, testing_vector[0], k)
    result = findMostOccur(list_neighbor)
    # print(result)
    return result

```

Hình 3.12: Dự đoán nhãn

3.5 Mô hình phân loại màu ảnh

File ảnh được truyền vào mô hình, các đặc trưng được trích xuất bằng cách sử dụng hàm color_histogram_of_test_image và đưa vào giải thuật KNN để phân loại màu sắc chủ đạo của ảnh.

```python
import cv2
from color_recognition_api import image_feature_extraction
from color_recognition_api import KNN
import os
import os.path
import sys

def sol(test_img_path):
# read the test image
    try:
        source_image = cv2.imread(sys.argv[1])
    except:
        source_image = cv2.imread(test_img_path)
    prediction = 'n.a.'

    image_feature_extraction.color_histogram_of_test_image(source_image)
    prediction = KNN.classify_img('./testing_dataset/testing_image.data')

    img = cv2.resize(source_image, (450,400))
    img = cv2.copyMakeBorder(img, 10, 50, 10, 10, borderType=cv2.BORDER_CONSTANT, value=(50, 50, 50))
    cv2.putText(img, "Prediction: " + prediction, (40, 450), cv2.FONT_HERSHEY_DUPLEX, 1.3, (255, 255, 255))
    cv2.imshow('image', img)
    cv2.waitKey()
    # Display the resulting frame
    cv2.waitKey(0)
```

Hình 3.13: Mô hình phân loại

3.6 Đánh giá mô hình

3.6.1 Độ chính xác

Mô hình được đánh giá theo tiêu chí độ chính xác (accuracy). Công thức tính độ chính xác như sau:

$$accuracy = \frac{\text{số lượng dự đoán đúng}}{\text{tổng số mẫu trong tập Test}}$$

Trong đó:

- Số lượng dự đoán đúng là tổng số lượng các dự đoán mà mô hình phân loại đúng.
- Tổng số mẫu trong tập dữ liệu là tổng số mẫu được sử dụng để đánh giá hiệu suất của mô hình

Với mỗi file ảnh, nếu nhãn dự đoán của chúng trùng khớp so với nhãn đã gán ban đầu, thì số lượng file dự đoán đúng sẽ được tăng 1 đơn vị. Mã giả của phần này trong hình 3.14

```python
def main(training_path, testing_path):
    '''
    Hàm này đánh giá model
    training_path: đường dẫn đến file lưu các đặc trưng của tập Training (đã gán nhãn)
    testing_path: đương dẫn đến file luu các đăch trưng của tập Test (chưa có nhãn)
    '''
    training_vector=[]
    testing_vector = []

    training_vector = loadData(training_path)
    testing_vector = loadData(testing_path)
    k = 5

    cnt =0
    prediction=[]
    for i in range(len(testing_vector)):
        list_neighbor = k_NearestNeighbors(training_vector, testing_vector[i], k)
        result = findMostOccur(list_neighbor)
        prediction.append(result)
        if result == testing_vector[i][-1]:
            cnt +=1
            print(str(i+1)+ ' '+ str(testing_vector[i]) + '-->'+result)
        else:
            print(str(i+1)+ ' '+ str(testing_vector[i]) + '-->'+result + '(Wrong)')
    # print(cnt)
    print('Acurency: '+ str(cnt/len(testing_vector) * 100) )
```

Hình 3.14: Đánh giá mô hình

Kết quả: Độ chính xác của mô hình là **97.4%**

```
Acurency: 97.39413680781759

PS D:\HocTap\code\BTL-TTNT\BTL_TTNT_Rework>
```

Hình 3.15: Độ chính xác

3.6.2 Lựa chọn giá trị k phù hợp

Lần lượt thử từng giá trị của k cho giải thuật KNN trong khoảng [5, 29], ta thu các kết quả:

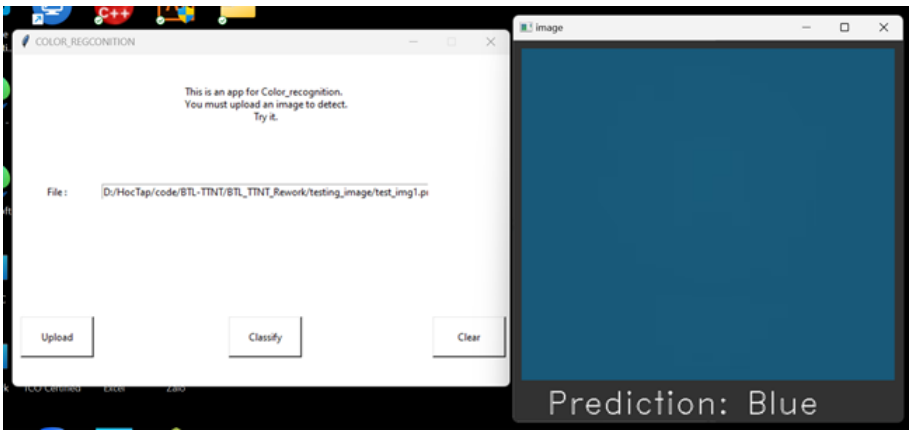
```
k=5 có Acurency: 96.41693811074919
k=6 có Acurency: 97.39413680781759
k=7 có Acurency: 96.74267100977198
k=8 có Acurency: 96.74267100977198
k=9 có Acurency: 96.74267100977198
k=10 có Acurency: 97.06840390879479
k=11 có Acurency: 96.74267100977198
k=12 có Acurency: 96.41693811074919
k=13 có Acurency: 95.43973941368078
k=14 có Acurency: 96.09120521172639
k=15 có Acurency: 96.09120521172639
k=16 có Acurency: 96.09120521172639
k=17 có Acurency: 95.76547231270358
k=18 có Acurency: 96.09120521172639
k=19 có Acurency: 96.74267100977198
k=20 có Acurency: 97.06840390879479
```

Hình 3.16: Độ chính xác của giải thuật KNN với mỗi giá trị K

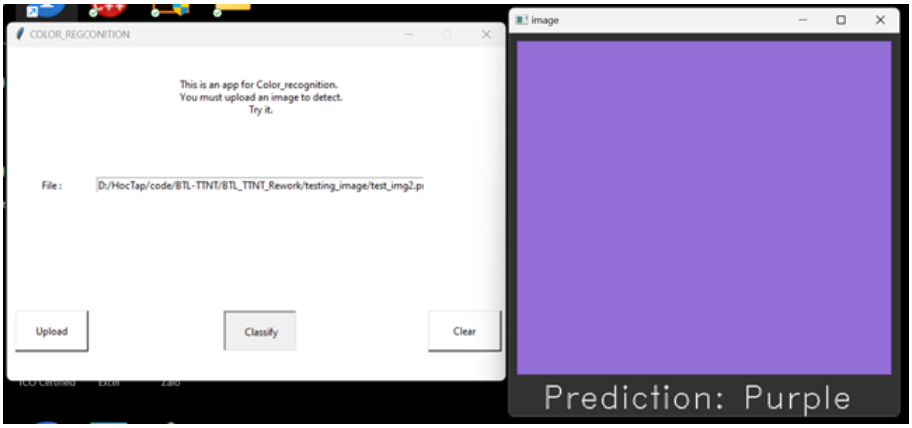
Trong hình 3.16, giá trị k=6 cho độ chính xác cao nhất mà giá trị này cũng không quá nhỏ cũng như không quá to, giúp giảm hiện tượng overfitting và giảm nhiễu khi phân loại. Bởi vậy, k=6 là một giá trị phù hợp với bài toán.

3.7 Sử dụng mô hình

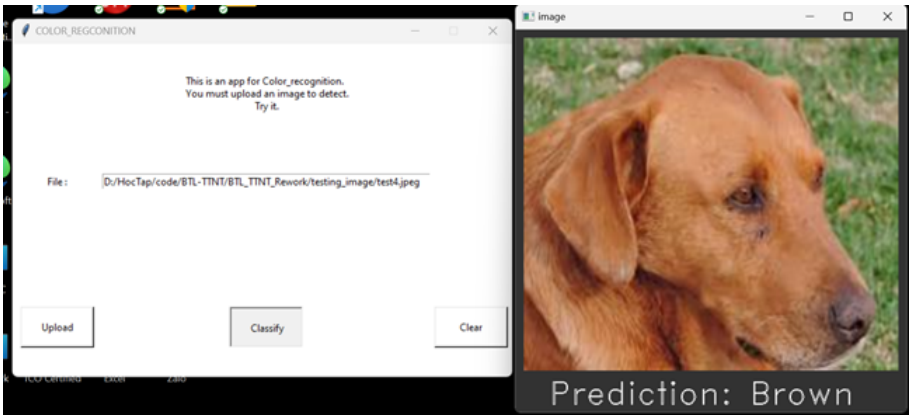
Để ứng dụng mô hình vào bài toán phân loại màu sắc, nhóm em tiến hành xây dựng một WinformApp, sau đó thực hiện dự đoán trên một số bức ảnh, cụ thể:



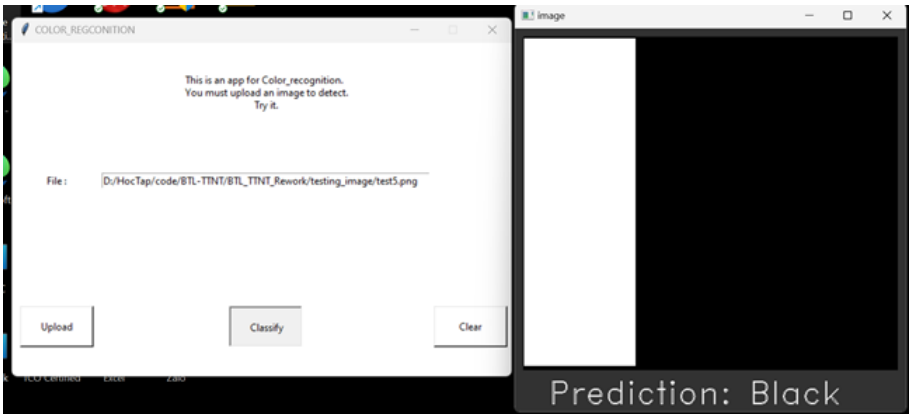
Hình 3.17: Dự đoán cho ảnh 1



Hình 3.18: Dự đoán cho ảnh 2



Hình 3.19: Dự đoán cho ảnh 3



Hình 3.20: Dự đoán cho ảnh 4

CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Cuối cùng, chương 4 chúng em sẽ đưa ra kết luận về bài toán phân loại màu ảnh sử dụng thuật toán KNNs, đồng thời cũng sẽ đề xuất một số hướng phát triển có thể có của bài toán và thuật toán này.

4.1 Kết luận

Tất cả những điều đã thảo luận về việc ứng dụng thuật toán K-Nearest Neighbors (KNN) cho bài toán phân loại màu ảnh cho thấy tầm quan trọng của việc nghiên cứu và áp dụng các thuật toán học máy trong xử lý ảnh. KNN không chỉ đơn thuần là một công cụ giúp phân loại màu sắc một cách hiệu quả mà còn mở ra những cơ hội mới trong các lĩnh vực khác của khoa học máy tính và trí tuệ nhân tạo.

Một điểm cần nhấn mạnh là việc hiểu rõ cách hoạt động của KNN trong việc tìm kiếm và phân loại dữ liệu. Thuật toán này hoạt động dựa trên nguyên lý "láng giềng gần nhất", giúp chúng ta phân loại màu sắc dựa trên các giá trị màu RGB của các điểm ảnh. Điều này phù hợp với nhiều loại bài toán khác nhau, từ nhận diện vật thể đến phân tích hình ảnh y khoa.

So với các phương pháp hiện đại như Mạng Nơron Tích Chập (CNN), KNN có độ chính xác thấp hơn nhưng lại có ưu điểm về sự đơn giản, dễ hiểu và tính dễ triển khai. Trong khi CNN yêu cầu nhiều tài nguyên tính toán và quá trình huấn luyện phức tạp, KNN lại không cần điều này và phù hợp với các bài toán đơn giản hơn.

Tuy nhiên, như mọi thuật toán khác, KNN cũng có nhược điểm của mình. Sự đơn giản và dễ triển khai của KNN có thể dẫn đến vấn đề hiệu suất khi xử lý các tập dữ liệu lớn và phức tạp. Đặc biệt, việc lựa chọn số lượng láng giềng k không phù hợp có thể làm giảm độ chính xác của thuật toán. Điều này yêu cầu chúng ta phải tinh chỉnh và tối ưu hóa k để đạt hiệu quả tốt nhất.

Việc ứng dụng thuật toán KNN cho bài toán phân loại màu ảnh mang lại hiệu quả nhất định, đặc biệt trong các trường hợp cần triển khai nhanh và không yêu cầu độ chính xác quá cao. Tuy nhiên, với những bài toán phức tạp và yêu cầu độ chính xác cao hơn, các phương pháp hiện đại như CNN sẽ là lựa chọn tốt hơn. Người dùng nên cân nhắc lựa chọn phương pháp phù hợp với yêu cầu cụ thể và tài nguyên hiện có để đạt được kết quả tối ưu.

Tóm lại, việc kết hợp kiến thức về không gian màu và thuật toán học máy như KNN không chỉ giúp chúng ta giải quyết các bài toán phân loại màu sắc một cách hiệu quả mà còn mở ra những cơ hội phát triển mới trong lĩnh vực này. Việc tiếp tục nghiên cứu và phát triển các thuật toán học máy không chỉ là hành động cần

thiết mà còn là chìa khóa để khám phá nhiều khía cạnh mới của trí tuệ nhân tạo và các lĩnh vực liên quan.

4.2 Hướng phát triển

Trong tương lai, nhóm em đặt mục tiêu phát triển sản phẩm của mình về việc sử dụng thuật toán KNN trong bài toán phân loại màu ảnh theo các hướng sau:

Đầu tiên là tối ưu hóa hiệu suất của thuật toán KNN. Nhóm sẽ tiếp tục nghiên cứu và áp dụng các kỹ thuật tối ưu hóa để giảm thời gian tính toán và tăng khả năng xử lý đối với các ảnh phức tạp và tập dữ liệu lớn. Việc này không chỉ cải thiện hiệu suất mà còn giúp sản phẩm trở nên linh hoạt và có khả năng xử lý nhiều tình huống khác nhau.

Tiếp theo là mở rộng áp dụng của thuật toán vào các lĩnh vực khác nhau. Nhóm quan tâm đến việc áp dụng đề tài này trong bài toán nhận diện màu sắc của một vật thể cụ thể, chẳng hạn như: phân loại sản phẩm theo màu sắc, phân loại các loại hoa,...Bên cạnh đó, nhóm cũng quan tâm đến ứng dụng thuật toán KNN trong bài toán phân loại văn bản chẳng hạn như phân loại email thành các nhóm spam hoặc không spam, hoặc phân tích cảm xúc của các bình luận trên mạng xã hội (tích cực, tiêu cực, trung lập). Điều này đòi hỏi nhóm phải nắm vững kiến thức về cả lĩnh vực đóng góp của mình và các lĩnh vực mở rộng để có thể tối ưu hóa giải pháp cho mỗi bài toán cụ thể.

Cuối cùng, nhóm dự định xây dựng một ứng dụng hoặc công cụ hỗ trợ sử dụng đề tài một cách linh hoạt và dễ dàng. Việc này không chỉ giúp tôi áp dụng kiến thức vào thực tiễn một cách hiệu quả mà còn chia sẻ và đóng góp cho cộng đồng lập trình và nghiên cứu trong lĩnh vực này.

4.3 Mức độ đóng góp

Trần Thị Thu Phương 26%

Mầu Nhân Tú 26%

Trần Việt Hà 16%

Nguyễn Hồng Đăng 16%

Đỗ Thị Quỳnh 16%