计算机科学与工程学院课程设计报告

题目全称： 图片聚类程序

题目难度等级： 4

指导老师： 蒲晓蓉 **职称：** 教授

学生姓名： 朱若愚 **学号：** 2022150501027

专业： 计算机科学与技术 **班号（大类分流后班号）：** 2022080913

A 计算机使用技能（100 分）	B 课程设计报告（100 分）	C 计算机专业技能（100 分）	总分（A*10%+B*30%+C*60%）

备注：如参加答辩，请答辩老师给出 C 计算机专业技能的分数。请指导教师给出除计算机专业技能以外的其他分数。

如参加答辩，请答辩老师签字： _____

主要任务：

该工具能够接收一组图片作为输入，并运用经典聚类算法 K-Means 算法对这些图片进行自动化的聚类分析，整个聚类过程在图形界面（GUI）中进行，通过可视化手段更好地理解和分析聚类结果。此外,该工具还允许用户将聚类后的图片数据导出到本地文件系统,便于后续的进一步研究或应用。

详细功能描述：

1. 创建 GUI 界面，展示程序窗口并初始化模型参数并选择预训练模型和图片文件夹准备进行图片聚类操作，进行特征提取，提取的特征用于后续的聚类分析，并实时显示加载进度。
2. 设置聚类参数，用户可输入聚类数量来指定聚类算法的参数，点击开始聚类后，程序会根据设置的参数和提取的图片特征进行聚类分析，并通过进度条和进度文本实时反馈聚类进度。
3. 展示聚类结果，聚类完成后，程序会在可滚动画布上以图片缩略图的形式展示聚类结果，用户可以直观地查看每个聚类簇中的图片，并可通过滚动条浏览所有聚类结果。
4. 保存聚类结果，点击保存聚类结果按钮，选择保存路径后，程序会将聚类结果保存到本地。

预期成果或目标：

给定一组图片对其使用经典聚类算法进行自动聚类;有 GUI 界面,指定聚类算法参数展示聚类结果。

指导老师评语：

指导教师签字： _____

摘　要

本课设项目旨在实现一个基于经典聚类算法的动物图片自动聚类系统，具备直观的 GUI 界面，支持用户灵活选择聚类参数并可视化展示聚类结果。项目采用 Python 语言开发，核心功能包括：通过预训练深度学习模型（如 EfficientNet、MobileNet 等）提取图片特征，利用主成分分析（PCA）对高维特征降维，再运用 K - Means 算法实现高效聚类。系统允许用户选择不同模型以适应质量、效率等不同需求场景，通过进度条实时反馈图片加载状态。聚类完成后，以滚动画布形式清晰展示各聚类簇的图片分布，并提供保存功能将结果分类存储至本地文件夹。该工具不仅适用于动物图片的分类整理，还可拓展至其他图片领域的自动聚类任务，为图像管理与分析提供了一种高效、便捷的解决方案，在图像数据库构建等场景中具有潜在应用价值。

关键词：K - Means 算法 图片聚类 GUI 页面 深度学习

目 录

第一章 绪 论

1.1 K – Means 算法介绍

1.1.1 K－Means 算法原理

K - Means 算法是一种广泛应用于数据分析和模式识别的无监督聚类算法，其核心目标是将数据集划分为 k 个簇，使得簇内数据点尽可能紧密聚集，而簇间数据点尽可能相互分离。具体而言，其工作原理基于迭代优化的过程来实现这一目标。

首先，K - Means 算法需要一个关键的输入参数 k，即期望的聚类数目。在初始化阶段，算法会随机地从数据集中选取 k 个数据点作为初始的簇心，这些簇心代表了各个簇的中心位置。接下来，在分配阶段，算法会计算数据集中每个数据点到各个簇心的距离，通常采用欧氏距离等度量方式。根据计算得到的距离，每个数据点会被分配到距离其最近的簇心中对应的簇中。

然后，进入更新阶段，算法会重新计算每个簇的簇心位置。新的簇心是通过取簇中所有数据点的坐标的均值得到的，这使得簇心能够更好地代表该簇数据点的中心位置，反映了簇内数据点的集中趋势。此时，簇心位置相较于初始化阶段已经发生了变化，基于新的簇心，再次执行分配阶段的操作，将数据点重新分配到距离最近的簇中。

上述的分配与更新过程会不断迭代重复，直到满足特定的停止条件。通常情况下，停止条件包括簇心位置的变化小于某个设定的阈值，或者达到预先设定的最大迭代次数。当迭代停止时，最终得到的簇划分结果就是 K - Means 算法的输出，每个簇由其簇心以及所属的数据点共同定义，从而实现了对数据集的聚类分析。

1.1.2 K－Means 算法应用

K - Means 算法凭借其简单高效的特点，在众多领域有着广泛的应用。以下是一些典型的应用场景：

在图像处理领域，K - Means 常用于图像分割。它可以将图像中的像素点划分为不同的簇，每个簇代表图像中的一个区域或对象。例如在医学影像分析中，通过 K - Means 聚类可以将不同组织或器官从图像中分离出来，便于医生进行诊断。同时，它还被应用于图像压缩和颜色量化。通过减少图像中颜色的种类，用少量颜色来近似表示原始颜色，从而降低图像存储空间，提高传输效率。

在市场营销方面，K - Means 算法能够对客户进行细分。企业可以根据客户的购买行为、消费偏好、人口统计学特征等多维度数据，将客户划分为不同的群体。针对不同群体的特点，企业可以制定个性化的营销策略，提高营销效果和客户满意度。比如，电商可以将客户分为高消费群体、中等消费群体和低消费群体，针对高消费群体提供专属优惠和服务，针对低消费群体推出入门级产品推荐。

在文本挖掘领域，K - Means 可用于文档聚类。它可以将大量的文本文档自动划分为若干主题相关的组别，方便信息的组织和检索。例如在新闻分类中，将新闻文章聚类为体育、娱乐、政治等类别，帮助用户快速找到自己感兴趣的新闻内容。此外，在生物信息学中，K - Means 也被用于基因表达数据分析等，协助研究人员发现基因表达模式和潜在的生物学意义。

1.2 图片聚类原理

图片聚类是一种将大量图片按照其内容、特征或相似性自动分组的技术，其核心在于通过提取图片的关键特征并利用聚类算法对这些特征进行分析，从而实现对图片的自动分类。在本项目中，我们主要采用基于深度学习特征提取与经典聚类算法相结合的方式进行图片聚类。

首先，图片聚类的关键步骤是特征提取。传统的图片特征提取方法依赖于手工设计的特征，如颜色直方图、纹理特征、形状特征等。然而，这些手工特征往往难以全面、准确地描述图片的复杂内容，且对不同类型的图片适应性较差。随着深度学习技术的发展，基于卷积神经网络（CNN）的自动特征提取方法逐渐成为主流。深度学习模型通过在大规模数据集上的预训练，能够学习到图片的多层次、语义丰富的特征表示。这些特征不仅包含了图片的低层视觉信息（如边缘、纹理等），还能够捕捉到高层语义信息（如物体类别、场景等），从而为后续的聚类分析提供了更强大、更具区分度的特征基础。

在得到图片的特征表示后，下一步就是应用聚类算法对这些特征进行分组。K - Means 算法作为一种简单而高效的聚类方法，在图片聚类任务中得到了广泛应用。它通过迭代优化的方式，将特征空间中的图片划分为若干个簇，使得同一簇内的图片在特征上具有较高的相似性，而不同簇之间的图片则具有较大的差异性。具体来说，K - Means 初始化 k 个簇心后，通过不断重复分配数据点到最近簇心对应的簇以及更新簇心位置这两个步骤，逐渐调整聚类结构，直至达到收敛条件。最终，每个簇中的图片在视觉内容或语义上具有一定的相似性，从而实现了对图片的自动分类。

第二章 系统分析

2.1 功能分析

2.1.1 功能概述

本项目所开发的动物图片聚类工具具备一系列功能，旨在为用户提供更便捷、高效的图片聚类体验。其核心功能涵盖模型选择与切换、图片文件夹选择、图片加载与特征提取、聚类参数设置、聚类操作执行、聚类结果显示与可视化以及聚类结果保存等。这些功能相互协作，共同实现了从图片输入到聚类结果输出的完整流程，满足用户对动物图片进行自动分类与组织的需求。

2.1.2 模型选择与切换功能

本工具为用户提供了多种预训练深度学习模型选项，包括 EfficientNetV2S、EfficientNetB0 和 MobileNetV3Small。用户可以通过界面中的模型选择菜单，根据自身的聚类需求（如质量优先、效率优先等）灵活切换不同的模型。当用户切换模型时，系统会相应地更新模型参数，确保后续的图片特征提取过程基于所选模型进行。

2.1.3 图片文件夹选择功能

用户可以点击 “选择图片文件夹” 按钮，从本地文件系统中浏览并选定包含待聚类动物图片的文件夹。该功能支持常见的图片格式，如.jpg、.png 和.jpeg。一旦选定文件夹，工具将自动读取其中的图片文件列表，为后续的图片加载和特征提取做好准备.

2.1.4 图片加载与特征提取功能

在选定图片文件夹后，工具会启动图片加载与特征提取过程。它会逐个读取文件夹中的图片，利用选定的深度学习模型对每张图片进行特征提取。在加载和处理图片的过程中，界面中的进度条和进度文本会实时更新，向用户反馈当前的加载进度，使用户能够清楚地了解系统正在执行的操作。

2.1.5 聚类参数设置功能

工具提供了灵活的聚类参数设置选项。用户可以在界面中输入期望的聚类数量，根据实际需求调整聚类结果的粒度。

2.1.6 聚类结果可视化及保存功能

聚类完成后，工具会在界面的可滚动画布区域以直观的方式展示聚类结果。

每个聚类簇会被单独显示，簇内包含的图片会以缩略图的形式排列展示。同时，还会显示每个聚类簇的编号以及该簇包含的图片数量等信息，方便用户快速了解聚类结果的分布情况。通过可视化的方式，用户可以轻松地浏览和比较不同聚类簇中的图片，评估聚类效果。

为了方便用户后续使用和分析聚类结果，工具提供了聚类结果保存功能。用户可以点击 “保存聚类结果” 按钮，选择目标文件夹，工具会将聚类后的图片按照所属的聚类簇分类复制到相应的子文件夹中。

2.2 流程关系分析

功能流程　如下图（图 2-2）所示

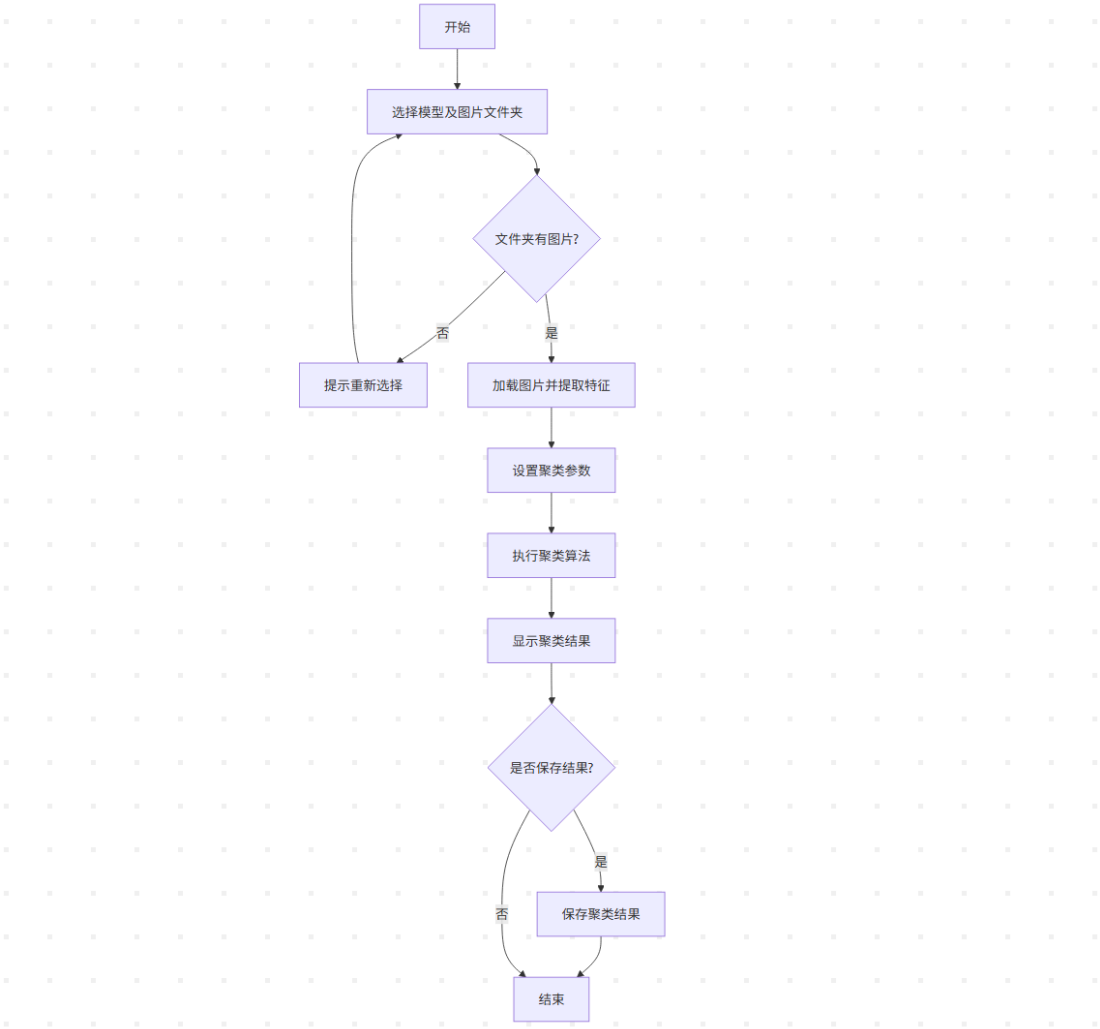


图 2-2 流程图

第三章 详细设计及实现

3.1 系统架构设计

本项目采用分层架构设计，主要分为前端 GUI 层、业务逻辑层和数据处理层。前端 GUI 层基于 ttkbootstrap 构建，负责与用户的交互，包括界面展示、用户输入接收等功能。业务逻辑层是系统的核心，处理图片加载、特征提取、聚类算法执行等主要任务。数据处理层则主要负责图片数据的预处理、特征存储和聚类结果的保存等操作。各层之间通过定义良好的接口进行通信，具有良好的解耦合性，便于后续的维护和扩展。

3.2 __init__ 函数

初始化应用程序，设置窗口标题、尺寸，并初始化关键变量，同时调用 create_widgets 方法创建 GUI 组件。

```
def __init__(self, root):
        self.root = root
        self.root.title("图片聚类工具")
        self.root.geometry("1920x1080")

        self.image_folder = None
        self.images = []
        self.features = []
        self.cluster_labels = []
        self.n_clusters = ttk.IntVar(value=2)
        self.model_name = ttk.StringVar(value="EfficientNetB0")  # 默认模型
        self.input_size = (224, 224)
        self.model = self.load_model(self.model_name.get())

        # 创建 GUI 组件
        self.create_widgets()
```

在 __init__ 方法中，首先将传入的 root 窗口设置为实例的属性，方便后续操作。接着设置窗口的标题为 “图片聚类工具”，并指定窗口的初始大小为 1920x1080 像素。然后初始化几个关键的实例变量：self.image_folder 用于存储用户选择的图片文件夹路径；self.images 列表用于存储图片文件的路径；self.features 列表用于存储提取的图片特征；self.cluster_labels 列表用于存储聚类结果标签。self.n_clusters 是一个整数变量，初始值设为 2，表示默认的聚类数量；self.model_name 是一个字符串变量，初始值设为 "EfficientNetB0"，表示默认选择的模型名称；self.input_size 元组用于存储模型输入图片的大小，默认设为 (224,

224)。最后调用 self.load_model 方法加载默认的模型，并调用 self.create_widgets 方法创建 GUI 界面的各个组件。

3.3 create_widgets 函数

创建 GUI 界面的所有组件，包括模型选择下拉菜单、按钮、进度条、标签等，并布局这些组件。

```
 def create_widgets(self):
        """
        创建应用程序的 GUI 组件，包括按钮、进度条和滚动画布。
        """
        # 模型选择下拉菜单
        model_label = ttk.Label(self.root, text="选择模型:")
        model_label.pack(pady=5)

        # 更新模型选项为带有描述的格式
        model_options = [
            "EfficientNetV2S(质量)",
            "EfficientNetB0(推荐)",
            "MobileNetV3Small(效率)"
        ]
        self.model_menu = ttk.Combobox(
            self.root, textvariable=self.model_name, values=model_options,
state="readonly"
        )
        self.model_menu.current(1)
        self.model_menu.bind("<<ComboboxSelected>>", lambda e:
self.switch_model(self.model_name.get()))
        self.model_menu.pack(pady=5)

        self.select_folder_button = ttk.Button(
            self.root, text="选择图片文件夹", command=self.select_folder,
bootstyle=PRIMARY
        )
        self.select_folder_button.pack(pady=10)

        # 显示加载进度的文本
        self.progress_text = ttk.Label(self.root, text="加载进度: 0/0")
        self.progress_text.pack(pady=5)

        # 创建进度条
        self.progress_bar = ttk.Progressbar(self.root, orient=HORIZONTAL,
length=600, mode="determinate")
        self.progress_bar.pack(pady=10)
```

```
        # 聚类数量输入框
        self.cluster_label = ttk.Label(self.root, text="聚类数量:")
        self.cluster_label.pack(pady=5)
        self.cluster_entry = ttk.Entry(self.root,
textvariable=self.n_clusters)
        self.cluster_entry.pack(pady=5)

        # “开始聚类”按钮
        self.cluster_button = ttk.Button(
            self.root, text="开始聚类", command=self.start_clustering,
bootstyle=DANGER
        )
        self.cluster_button.pack(pady=10)

        self.create_scrollable_canvas()

        self.save_button = ttk.Button(
            self.root, text="保存聚类结果", command=self.save_results,
bootstyle=DARK
        )
        self.save_button.pack(pady=10)
```

create_widgets 方法通过一系列的 GUI 组件创建和布局操作，构建了完整的用户界面。首先创建了一个标签 model_label，用于显示 “选择模型:” 的提示文本，并将其放置在窗口中。接着创建了一个下拉菜单 self.model_menu，使用 ttk.Combobox 组件，将 self.model_name 变量与之绑定，并设置可选的模型列表。当用户选择不同的模型时，触发绑定的事件处理函数，调用 self.switch_model 方法进行模型切换。之后创建了 “选择图片文件夹” 按钮，点击时触发 self.select_folder 方法。再创建进度文本标签 self.progress_text 和进度条 self.progress_bar，用于显示图片加载进度。然后创建了聚类数量标签 self.cluster_label 和输入框 self.cluster_entry，允许用户输入聚类数量。随后创建了 “开始聚类” 按钮，点击时触发 self.start_clustering 方法。调用 self.create_scrollable_canvas 方法创建可滚动画布，用于展示聚类结果。最后创建了 “保存聚类结果” 按钮，点击时触发 self.save_results 方法。

3.4 create_scrollable_canvas 函数

创建一个可滚动的画布组件，用于展示聚类结果。

```
def create_scrollable_canvas(self):
        container = ttk.Frame(self.root)
        container.pack(fill=BOTH, expand=True)
```

```
        self.scroll_canvas = ttk.Canvas(container,
width=self.root.winfo_width() - 20, height=400)
        self.scroll_canvas.pack(side=LEFT, fill=BOTH, expand=True)

        scrollbar = ttk.Scrollbar(container, orient=VERTICAL,
command=self.scroll_canvas.yview)
        scrollbar.pack(side=RIGHT, fill=Y)

        self.scroll_canvas.configure(yscrollcommand=scrollbar.set)

        self.scrollable_frame = ttk.Frame(self.scroll_canvas)
        self.scrollable_frame.bind(
            "<Configure>",
            lambda e:
self.scroll_canvas.configure(scrollregion=self.scroll_canvas.bbox("all"))
        )

        self.scroll_canvas.create_window((0, 0),
window=self.scrollable_frame, anchor="nw")

        # 绑定鼠标滚轮事件
        self.scroll_canvas.bind_all("<MouseWheel>", self._on_mouse_wheel)

    def _on_mouse_wheel(self, event):
        self.scroll_canvas.yview_scroll(-1 * (event.delta // 120), "units")
```

create_scrollable_canvas 方法首先创建了一个容器框架 container，用于放置画布和滚动条。然后创建了一个 ttk.Canvas 组件 self.scroll_canvas，设置其宽度为窗口宽度减 20 像素，高度为 400 像素。接着创建了一个垂直滚动条 scrollbar，并将其与画布关联，使得可以通过滚动条来查看画布中的内容。将滚动条放置在容器的右侧，并设置为垂直填充。然后创建了一个可滚动的框架 self.scrollable_frame，用于放置聚类结果的图片展示。为该框架绑定了配置事件，当框架大小改变时，调整画布的滚动区域。最后将可滚动框架添加到画布中，并绑定鼠标滚轮事件到画布上，以便用户可以通过滚轮来滚动查看内容。

3.5 load_model 函数和 switch_model 函数

根据用户选择的模型名称，加载对应的预训练深度学习模型，并设置模型的输入图片大小。当用户在 GUI 界面中切换模型选择时，根据用户选择的新模型名称，重新加载对应的预训练模型，并更新系统的输入图片大小等参数。

```
def load_model(self, model_name):
        """
```

```
        根据模型名称加载对应的预训练模型，并设置输入大小。
        """
        if model_name == "EfficientNetV2S":
            self.input_size = (300, 300)
            return tf.keras.applications.EfficientNetV2S(
                weights="imagenet", include_top=False, pooling="avg",
input_shape=(300, 300, 3)
            )
        elif model_name == "EfficientNetB0":
            self.input_size = (224, 224)
            return tf.keras.applications.EfficientNetB0(
                weights="imagenet", include_top=False, pooling="avg",
input_shape=(224, 224, 3)
            )
        elif model_name == "MobileNetV3Small":
            self.input_size = (224, 224)
            return tf.keras.applications.MobileNetV3Small(
                weights="imagenet", include_top=False, pooling="avg",
input_shape=(224, 224, 3)
            )
        else:
            raise ValueError("不支持的模型名称")

    def switch_model(self, selected_model):
        """
        切换模型时重新加载对应的预训练模型，并更新输入大小。
        """
        actual_model_name = selected_model.split("(")[0]
        self.model = self.load_model(actual_model_name)
        messagebox.showinfo("信息", f"切换到模型: {selected_model}，请选择图
片文件夹")
```

load_model 方法接收一个参数 model_name，表示要加载的模型名称。根据不同的 model_name，使用 tf.keras.applications 模块加载相应的模型。对于 "EfficientNetV2S" 模型，设置输入大小为 (300, 300)，并加载预训练的 EfficientNetV2S 模型，去掉顶部分类层，添加平均池化层，指定输入形状为 (300, 300, 3)。对于 "EfficientNetB0" 模型，设置输入大小为 (224, 224)，并加载预训练的 EfficientNetB0 模型，同样去掉顶部分类层，添加平均池化层，指定输入形状为 (224, 224, 3)。对于 "MobileNetV3Small" 模型，设置输入大小为 (224, 224)，加载预训练的 MobileNetV3Small 模型，去掉顶部分类层，添加平均池化层，指定输入形状为 (224, 224, 3)。如果传入的 model_name 不是上述三种之一，则抛出 ValueError 异常，提示不支持该模型名称。switch_model 方法接收一个参数

selected_model，表示用户在 GUI 界面中选择的新模型。从 selected_model 中提取实际的模型名称（去掉可能附带的描述信息），然后调用 self.load_model 方法加载该模型。加载模型完成后，使用 messagebox.showinfo 方法弹出一个信息框，向用户提示已成功切换到新的模型。

3.6 select_folder 函数

允许用户选择包含待聚类图片的文件夹，并读取该文件夹中的图片文件列表。

```
def select_folder(self):
        self.image_folder = filedialog.askdirectory()
        if self.image_folder:
            self.images = []
            self.features = []

            # 获取文件列表
            image_files = [f for f in os.listdir(self.image_folder) if
f.endswith((".jpg", ".png", ".jpeg"))]
            total_files = len(image_files)

            if total_files == 0:
                messagebox.showinfo("信息", "文件夹中没有图片")
                return

            # 配置进度条
            self.progress_bar["maximum"] = total_files
            self.progress_bar["value"] = 0
            self.progress_text.config(text=f"加载进度: 0/{total_files}")

            for i, filename in enumerate(image_files):
                img_path = os.path.join(self.image_folder, filename)
                img = tf.keras.preprocessing.image.load_img(img_path,
target_size=self.input_size)  # 使用动态输入大小
                img_array = tf.keras.preprocessing.image.img_to_array(img)
                img_array =
tf.keras.applications.efficientnet.preprocess_input(img_array)
                img_array = tf.expand_dims(img_array, axis=0)

                feature = self.model.predict(img_array)
                self.features.append(feature.flatten())
                self.images.append(img_path)

                # 更新进度条和文本
                self.progress_bar["value"] = i + 1
```

```
                self.progress_text.config(text=f"加载进度: {i +
1}/{total_files}")
                self.root.update()  # 刷新 UI

            messagebox.showinfo("信息", f"已加载 {len(self.images)} 张图片,
请输入聚类数量")
```

select_folder 方法使用 filedialog.askdirectory 方法弹出文件夹选择对话框，让用户从本地文件系统中选择一个文件夹。将用户选择的文件夹路径存储在 self.image_folder 属性中。获取选定文件夹中的文件列表，筛选出以 .jpg、.png 和 .jpeg 为后缀的图片文件。统计图片文件的数量，如果数量为 0，则弹出信息框提示用户文件夹中没有图片，并清空 self.image_folder 属性，要求用户重新选择文件夹。如果文件夹中有图片文件，则初始化 self.images 和 self.features 列表，用于存储图片路径和提取的特征。然后开始加载图片并提取特征的过程，同时更新进度条和进度文本，显示加载进度。

3.7 start_clustering 函数

根据用户设置的聚类参数，调用聚类算法对提取的图片特征进行聚类分析，并将聚类结果存储以便后续展示和保存。

```
def start_clustering(self):
        if not self.images:
            messagebox.showerror("错误", "请先选择图片文件夹")
            return

        n_clusters = self.n_clusters.get()
        if n_clusters <= 0:
            messagebox.showerror("错误", "聚类数量必须大于 0")
            return

        n_samples = len(self.features)
        n_features = len(self.features[0]) if self.features else 0
        n_components = min(50, n_samples, n_features)

        if n_components <= 0:
            messagebox.showerror("错误", "数据不足以进行降维")
            return

        pca = PCA(n_components=n_components, random_state=42)
        reduced_features = pca.fit_transform(self.features)

        kmeans = KMeans(
```

```
            n_clusters=n_clusters, init='k-means++', n_init=20,
max_iter=5000,
            tol=1e-5, random_state=42, algorithm='lloyd'
        )
        self.cluster_labels = kmeans.fit_predict(reduced_features)

        self.display_clusters()
```

start_clustering 方法首先检查 self.images 是否为空，如果为空则弹出错误框提示用户先选择图片文件夹。获取用户输入的聚类数量 n_clusters，验证其是否大于 0，若不大于 0 则弹出错误框提示用户聚类数量必须大于 0。计算图片特征的样本数量 n_samples 和每个特征的维度 n_features。确定 PCA 降维的目标维度 n_components，取 n_samples、n_features 和 50 中的最小值。使用 sklearn.decomposition.PCA 进行 PCA 降维，将图片特征降维到 n_components 维。然后初始化 KMeans 模型，设置聚类数量为 n_clusters，采用 'k-means++' 初始化方法，设置 20 次初始化、5000 次最大迭代、容忍度为 1e-5，随机种子为 42，使用 'lloyd' 算法。对降维后的特征数据进行聚类训练，得到每个图片的聚类标签，存储在 self.cluster_labels 属性中。聚类完成后，调用 self.display_clusters 方法展示聚类结果。

3.8 display_clusters 函数及 save_results 函数

在 GUI 界面的可滚动画布上展示聚类结果，将同一聚类簇的图片展示在一起，并显示每个簇的编号和包含的图片数量。并将聚类结果保存到本地文件系统，按照聚类簇将图片分类复制到相应的子文件夹中。

```
def display_clusters(self):
        for widget in self.scrollable_frame.winfo_children():
            widget.destroy()

        cluster_images = [[] for _ in range(self.n_clusters.get())]

        for img_path, label in zip(self.images, self.cluster_labels):
            cluster_images[label].append(img_path)

        img_width, img_height = 100, 100
        padding_x, padding_y = 10, 10

        # 动态计算最大列数
        canvas_width = self.scroll_canvas.winfo_width()
```

```
        max_columns = max(1, ((canvas_width - padding_x*2) // (img_width +
padding_x))-2)

        y_offset = 0
        total_images = len(self.images)  # 总图片数量
        for i, cluster in enumerate(cluster_images):
            cluster_count = len(cluster)  # 当前类别的图片数量
            label = ttk.Label(
                self.scrollable_frame,
                text=f"聚类 {i+1} ({cluster_count}/{total_images})",
                anchor="w"
            )
            label.grid(row=y_offset, column=0, sticky="w", pady=(0, 10),
columnspan=max_columns)
            y_offset += 1

            x_offset = 0
            for j, img_path in enumerate(cluster):
                img = Image.open(img_path)
                img = img.resize((img_width, img_height),
Image.Resampling.LANCZOS)
                photo = ImageTk.PhotoImage(img)

                img_label = ttk.Label(self.scrollable_frame, image=photo)
                img_label.image = photo
                img_label.grid(row=y_offset, column=x_offset,
padx=padding_x, pady=padding_y)

                x_offset += 1
                if x_offset >= max_columns:
                    x_offset = 0
                    y_offset += 1

            y_offset += 1

    def save_results(self):
        if len(self.cluster_labels) == 0:
            messagebox.showerror("错误", "请先执行聚类")
            return

        save_folder = filedialog.askdirectory()
        if not save_folder:
            messagebox.showerror("错误", "未选择保存文件夹")
            return

        cluster_result_folder = os.path.join(save_folder, "cluster_result")
```

```
        if not os.path.exists(cluster_result_folder):
            os.makedirs(cluster_result_folder)

        for i in range(self.n_clusters.get()):
            cluster_folder = os.path.join(cluster_result_folder,
f"cluster_{i+1}")
            if not os.path.exists(cluster_folder):
                os.makedirs(cluster_folder)

            for img_path, label in zip(self.images, self.cluster_labels):
                if label == i:
                    shutil.copy(img_path, cluster_folder)

        messagebox.showinfo("信息", "聚类结果已保存")
```

display_clusters 方法首先清空可滚动画布上的所有现有内容。创建一个列表 cluster_images，其中每个元素是一个列表，用于存储对应聚类簇的图片路径。根据 self.cluster_labels 和 self.images，将图片路径分配到对应的聚类簇列表中。计算图片展示的宽度、高度和填充间距，动态确定每行展示的最大列数。然后为每个聚类簇创建一个标签，显示簇的编号和包含的图片数量。接着将每个聚类簇的图片以缩略图的形式在画布上展示，使用 ttk.Label 组件显示图片，并设置网格布局参数进行排列。在展示完所有聚类簇后，更新画布的滚动区域，以便用户可以通过滚动条查看所有内容。

save_results 方法首先检查 self.cluster_labels 是否为空，如果为空则弹出错误框提示用户先执行聚类。使用 filedialog.askdirectory 方法弹出文件夹选择对话框，让用户选择保存聚类结果的目标文件夹。如果用户没有选择文件夹，则弹出错误框提示用户未选择保存文件夹。在目标文件夹下创建一个名为 "cluster_result" 的子文件夹，用于存储聚类结果。然后根据聚类数量创建对应的子文件夹（如 cluster_1、cluster_2 等）。遍历图片路径和聚类标签，将每张图片复制到对应的聚类簇子文件夹中。最后弹出信息框提示用户聚类结果已保存。

第四章 测试

4.1 环境的构建

```
import os
import shutil
import ttkbootstrap as ttk
from ttkbootstrap.constants import *
from PIL import Image, ImageTk
import tensorflow as tf
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

from tkinter import filedialog, messagebox
```

1. import os：导入 Python 内置的 os 模块，用于操作系统相关的功能，如文件和目录操作、路径处理等。在本项目中，用到它来检查文件夹路径、创建目录等。

2. import shutil：导入 shutil 模块，它是一组用于文件操作的高级接口，如复制、移动、删除文件和目录等。在项目中用于保存聚类结果时复制图片到指定文件夹。

3. import ttkbootstrap as ttk：导入 tktbootstrap 库，并将其别名为 ttk。ttkbootstrap 是一个基于 Tkinter 的 GUI 库，为 Tkinter 提供了现代的、美观的界面组件，用于构建项目的图形用户界面 GUI。

4. from ttkbootstrap.constants import *：从 tktbootstrap.constants 模块导入所有的常量。用于设置 GUI 组件的样式和主题，如颜色、字体等。

5. from PIL import Image, ImageTk：从 Python Imaging Library（PIL）导入 Image 和 ImageTk 模块。Image 用于打开、操作和保存多种格式的图像文件，ImageTk 则用于将 PIL 图像转换为 Tkinter 可显示的格式。

6. import tensorflow as tf：导入 TensorFlow 库，并将其别名为 tf。TensorFlow 是一个用于机器学习和深度学习的开源框架，在本项目中用于加载预训练的深度学习模型进行图片特征提取。

7. from sklearn.cluster import KMeans：从 scikit-learn 库的 cluster 模块导入 KMeans 类。KMeans 是一种常用的聚类算法，用于对数据进行分组，这里是图片特征的聚类。

8. from sklearn.decomposition import PCA：从 scikit-learn 库的 decomposition 模块导入 PCA 类。主成分分析（PCA）用于减少数据的维度，同时保留大部分重要信息。

9. from tkinter import filedialog, messagebox：从 tkinter 库导入 filedialog 和 messagebox 模块。filedialog 用于创建打开和保存文件的对话框，messagebox 用于显示各种消息框，如错误提示、信息提示等。

4.2 GUI 界面显示

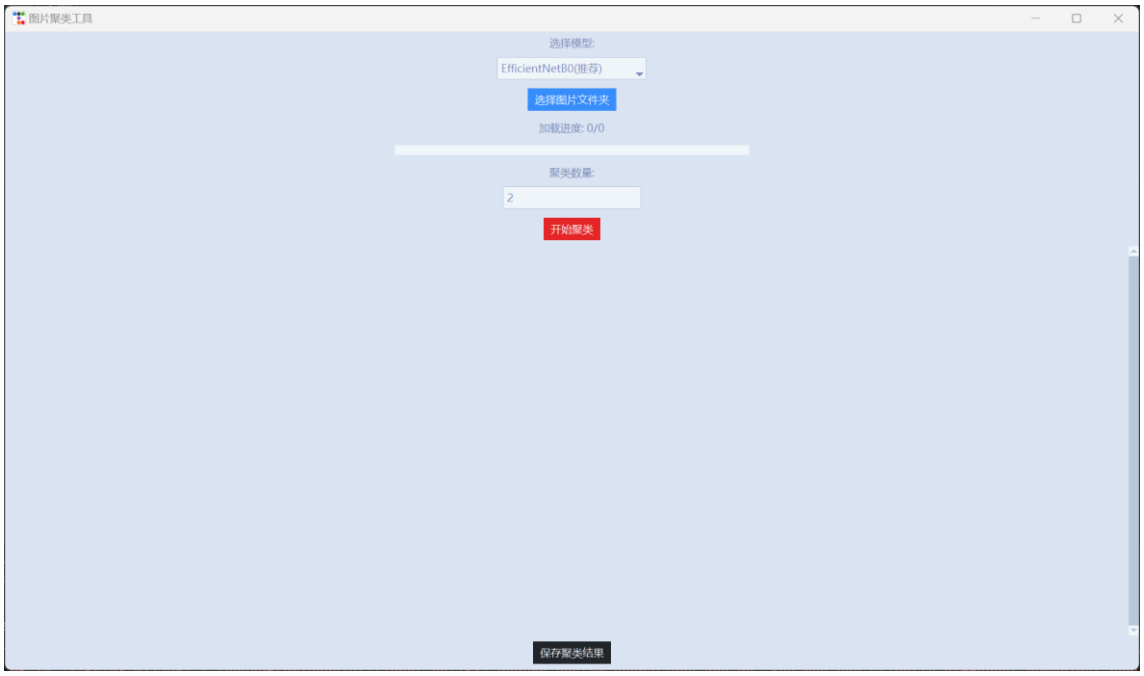


图 4-2 GUI 界面

4.3 6 张 2 类高区分度图片测试以及保存测试

FISH_HORSE 图片文件夹包含 6 张图片，其中鱼和马各 6 张：

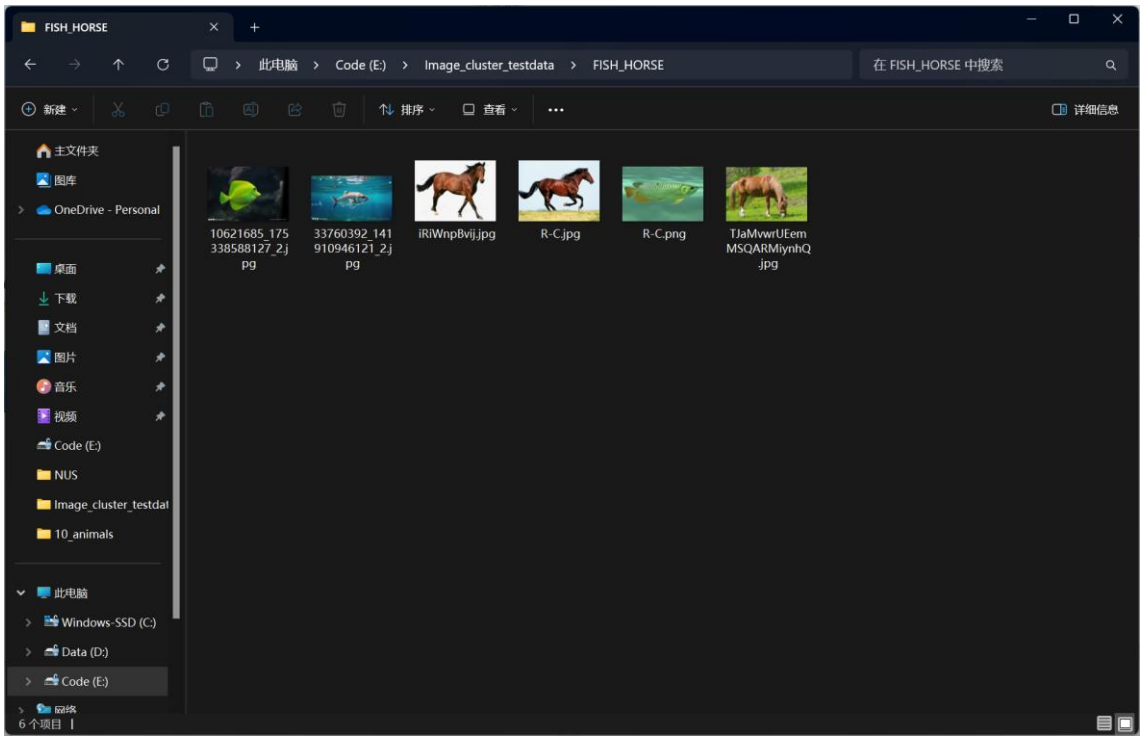


图 4-3-1FISH_HORSE 文件夹

以默认模型聚类数量 2 进行聚类：

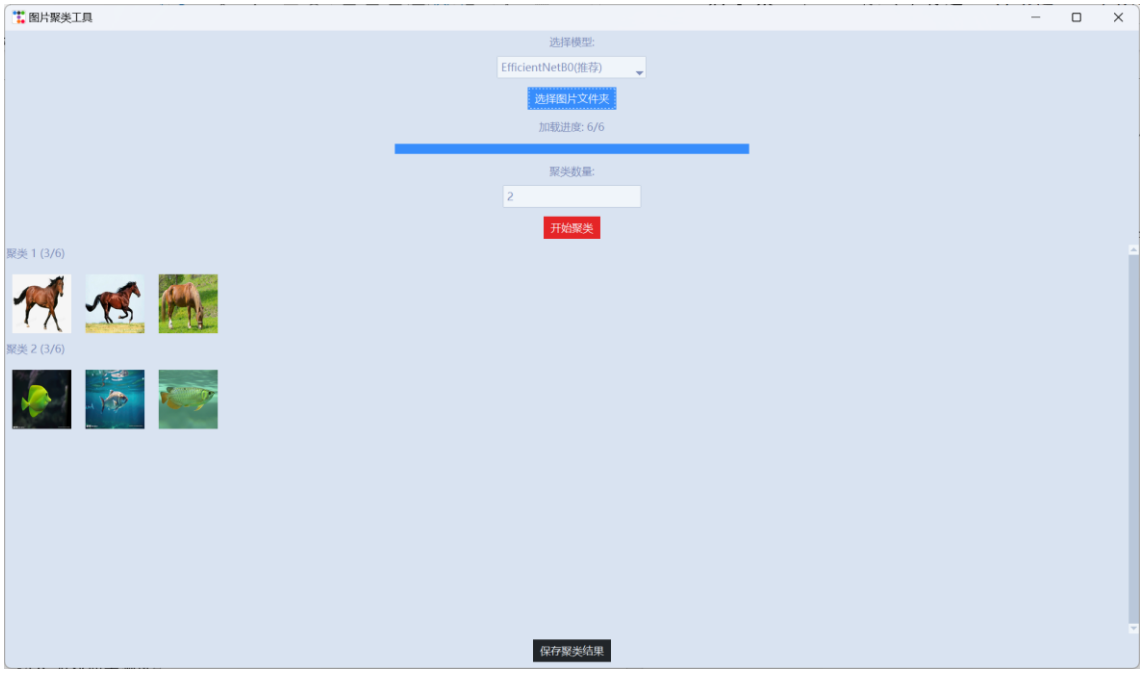


图 4-3-2 FISH_HORSE 聚类结果

可见正确聚类两种类型的图片。

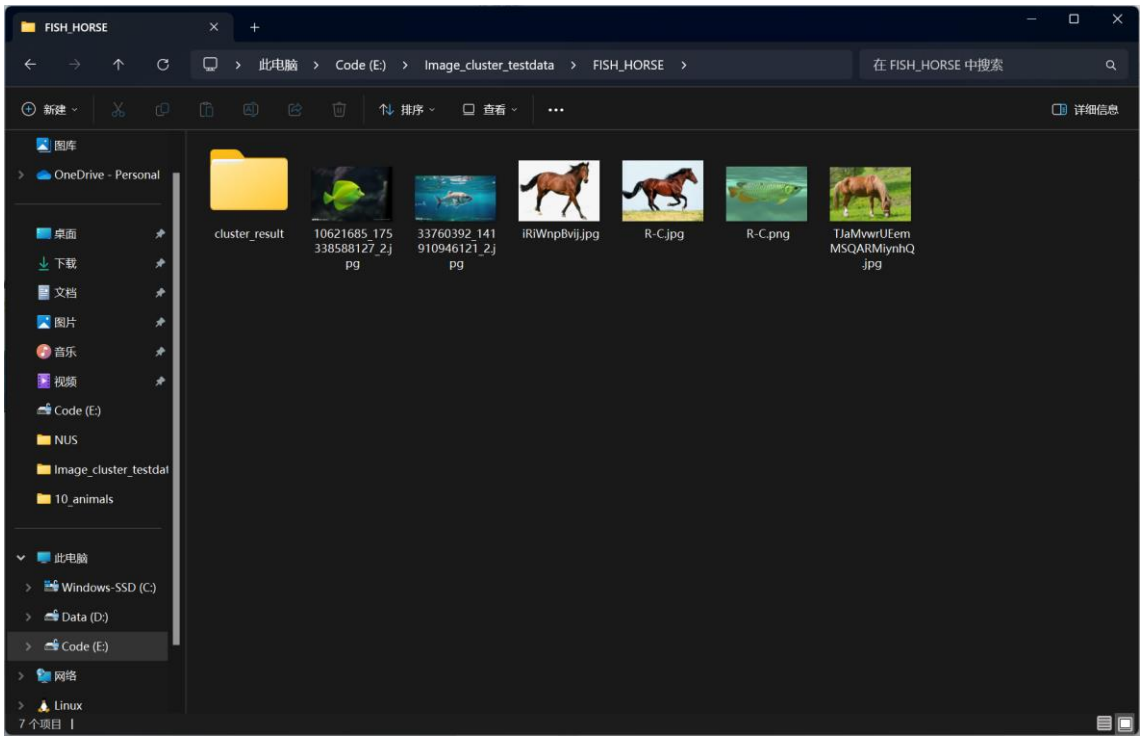


图 4-3-3 聚类结果保存（1）

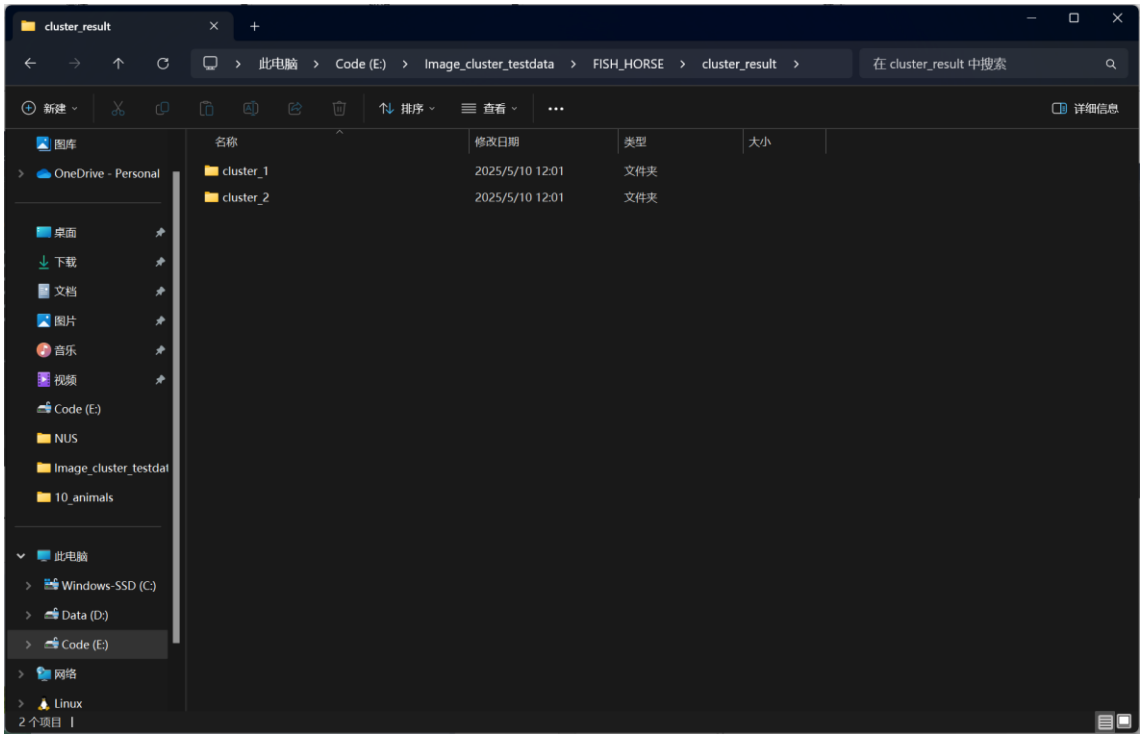
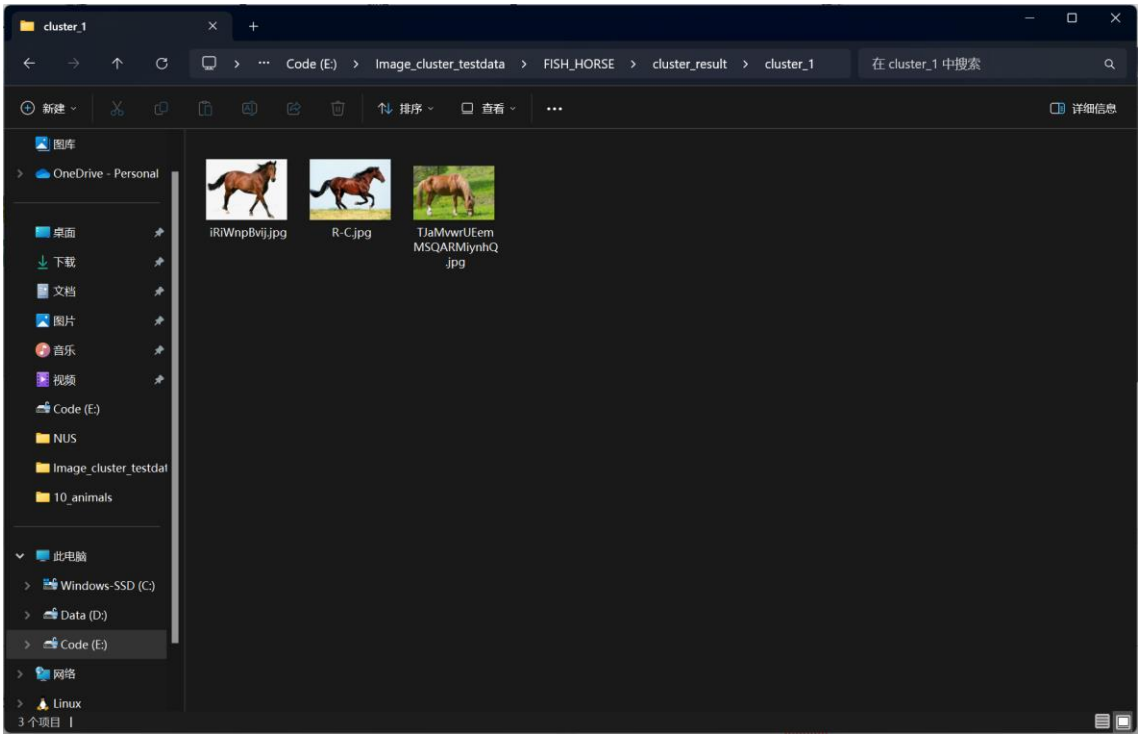
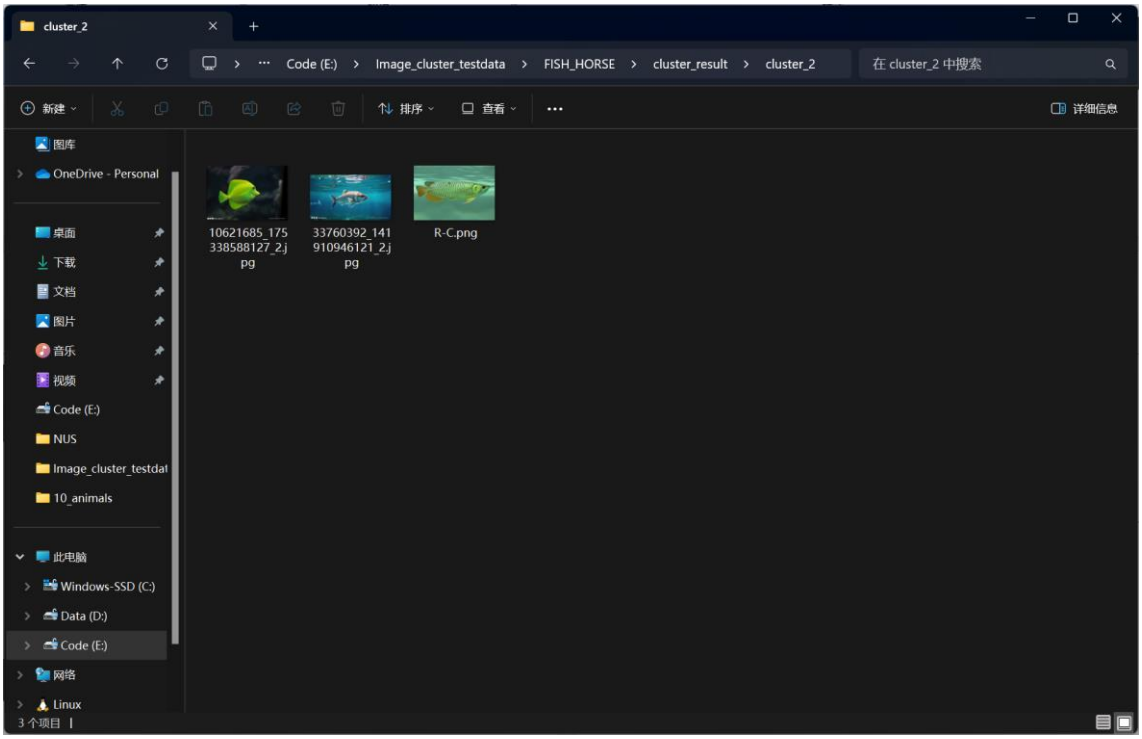


图 4-3-4 聚类结果保存（2）



4-3-5 聚类结果保存（3）



4-3-6 聚类结果保存（4）

4.4 100 张 2 类低区分度图片测试

CAT_DOG 图片文件夹包含 100 张图片，其中猫和狗各 50 张。

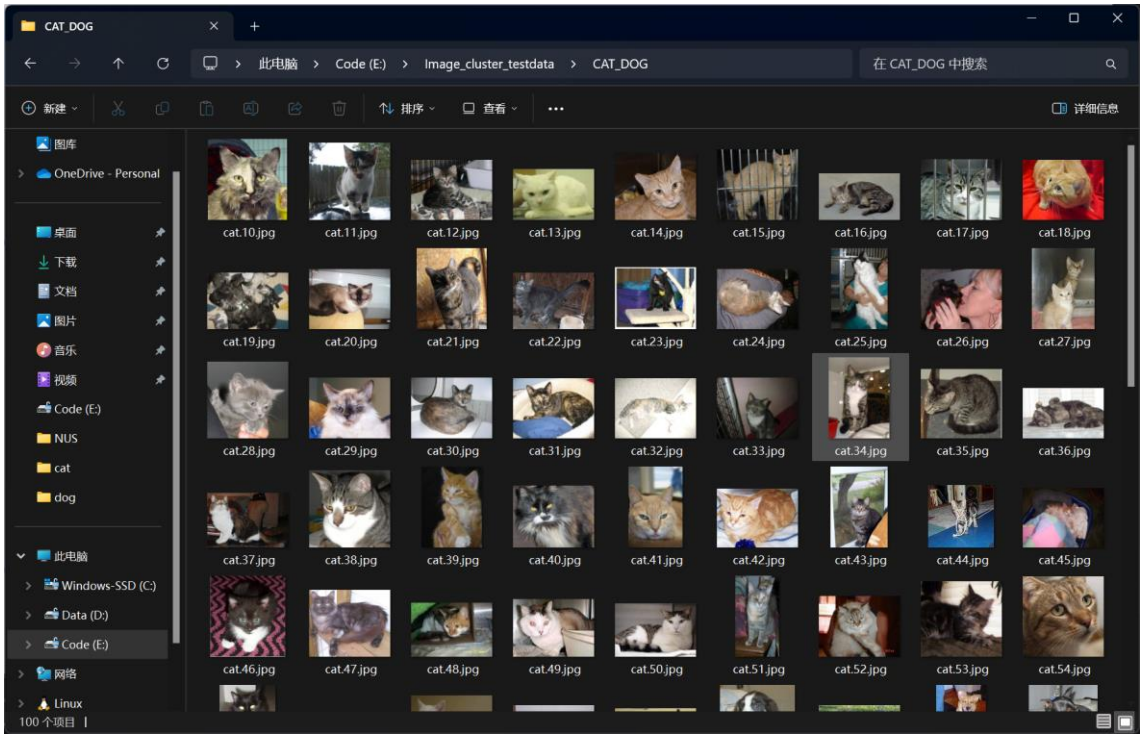
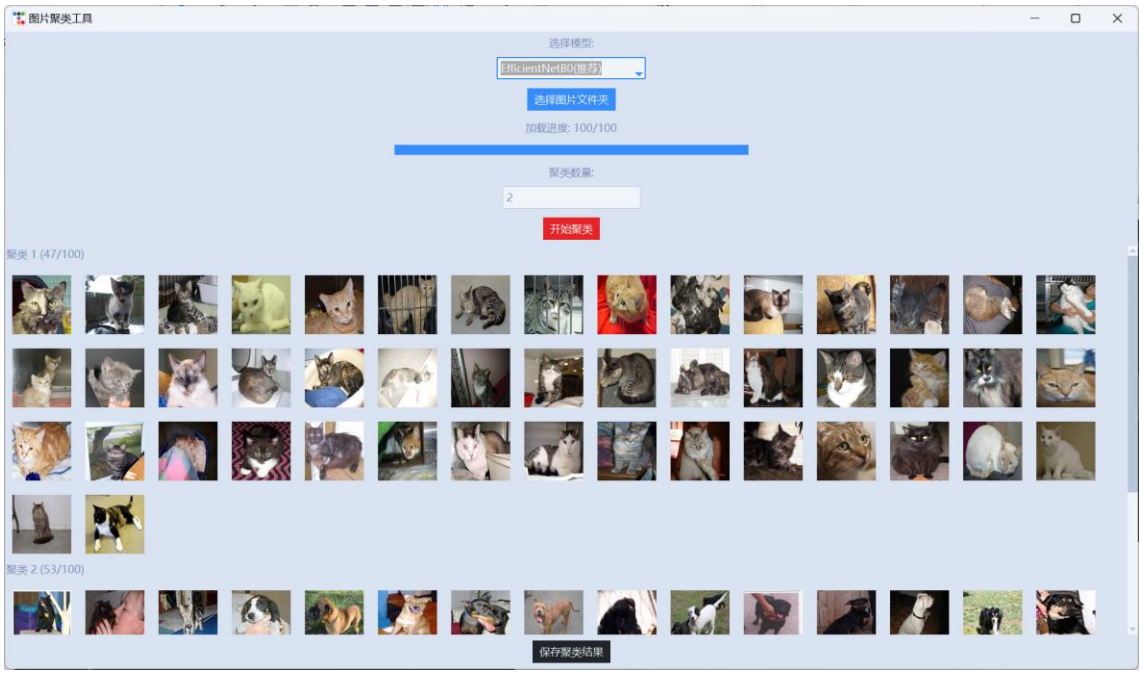


图 4-4-1 CAT_DOG 文件夹

分别采用三种模型进行测试：



4-4-2 EfficientNetB0 模型测试

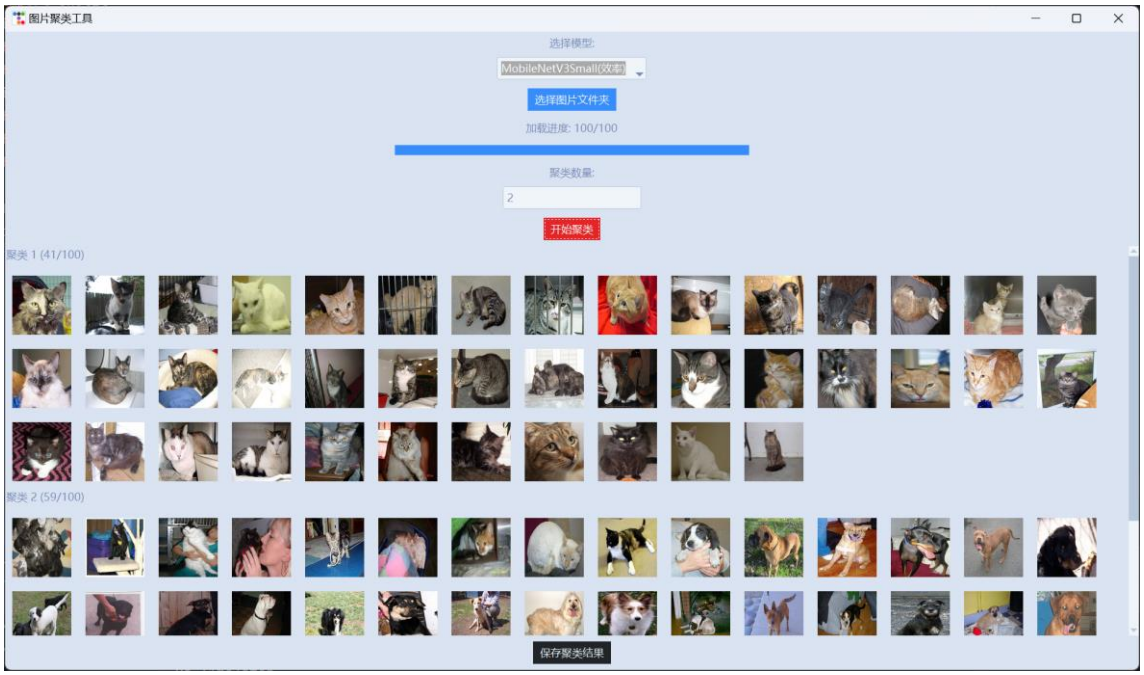


图 4-4-3 MobileNetV3Small 模型测试

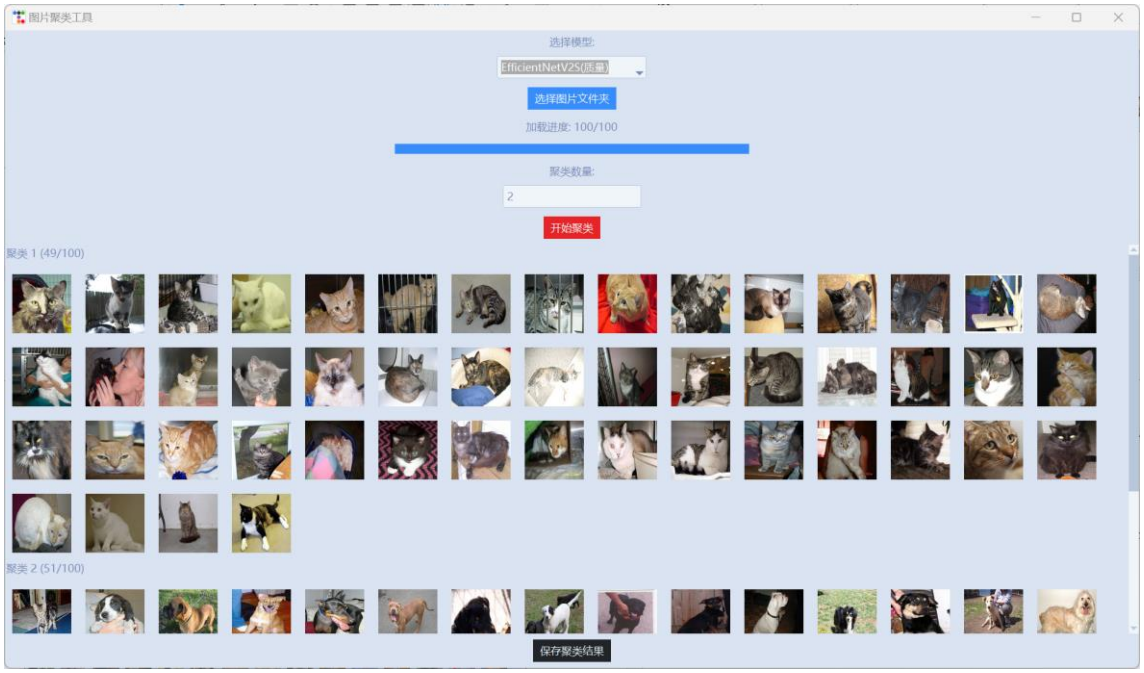


图 4-4-4 EfficientNetV2S 模型测试

测试结果如下表所示：

模型	平均加载速度（ms/张）	正确样本数	准确率
MobileNetV3Small（效率）	约 31	91	91%
EfficientNetB0（推荐）	约 44	97	97%
EfficientNetV2S（质量）	约 94	99	99%

表 4-4 100 张 2 类低区分度图片测试结果

4.5 1000 张 10 类中区分度图片测试

10_animals 图片文件夹包含 1000 张图片，其中 10 类动物各 100 张

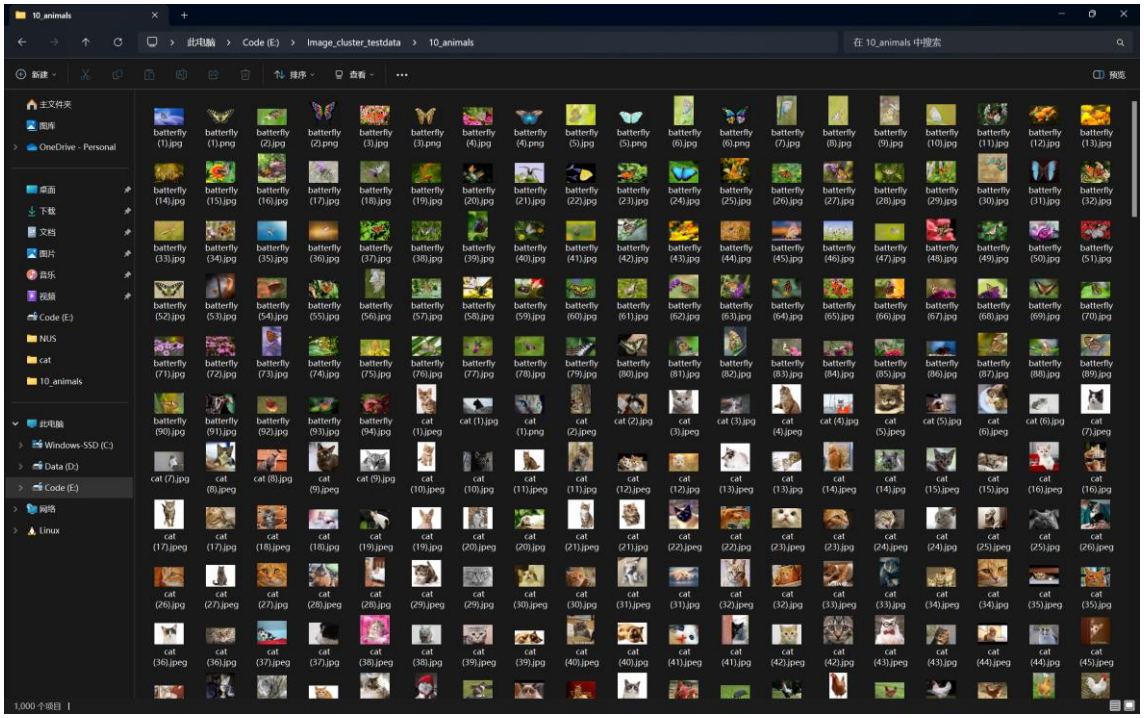
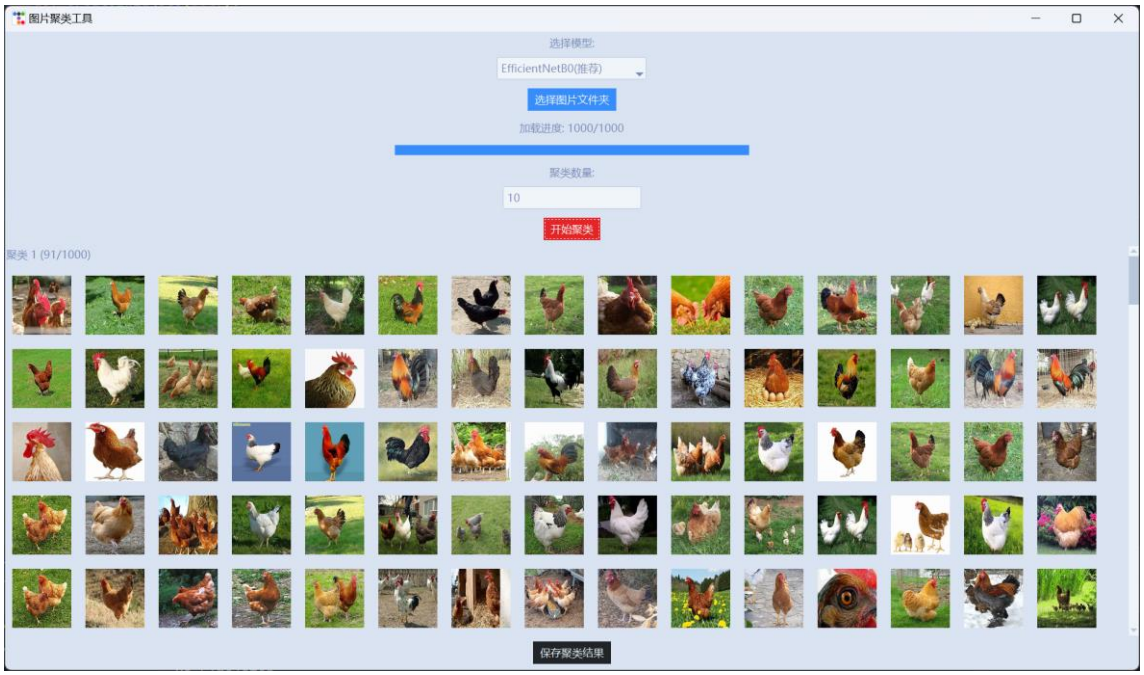


图 4-5-1 10_animals 文件夹

分别采用三种模型进行测试：



4-5-2 EfficientNetB0 模型测试

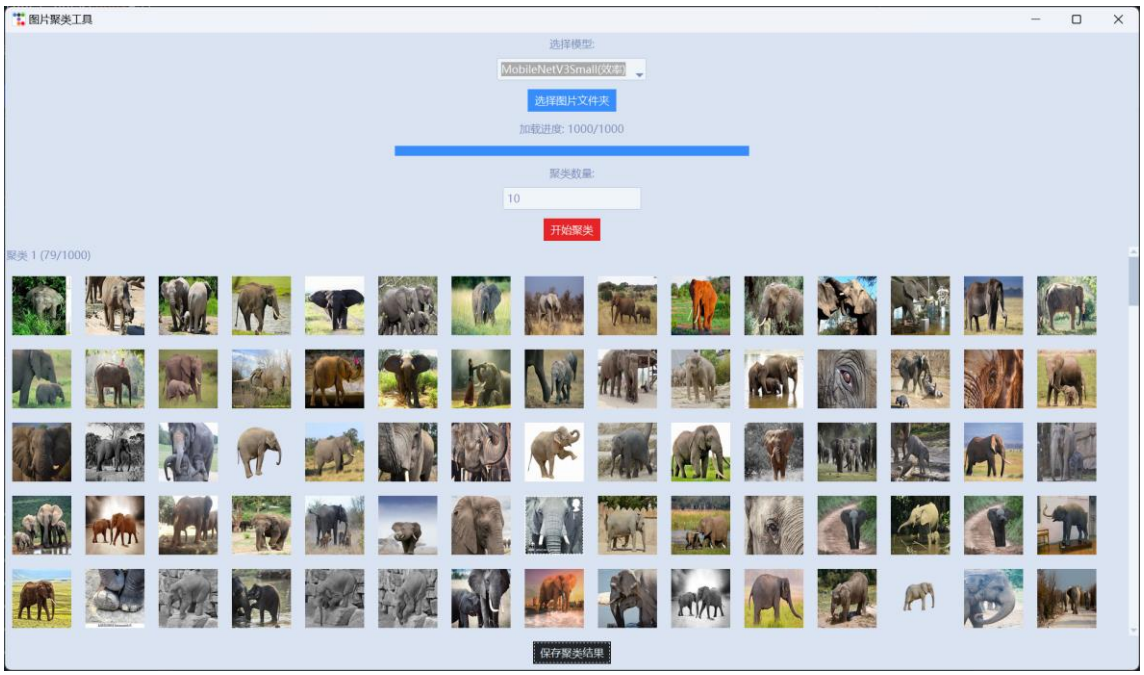


图 4-5-3 MobileNetV3Small 模型测试

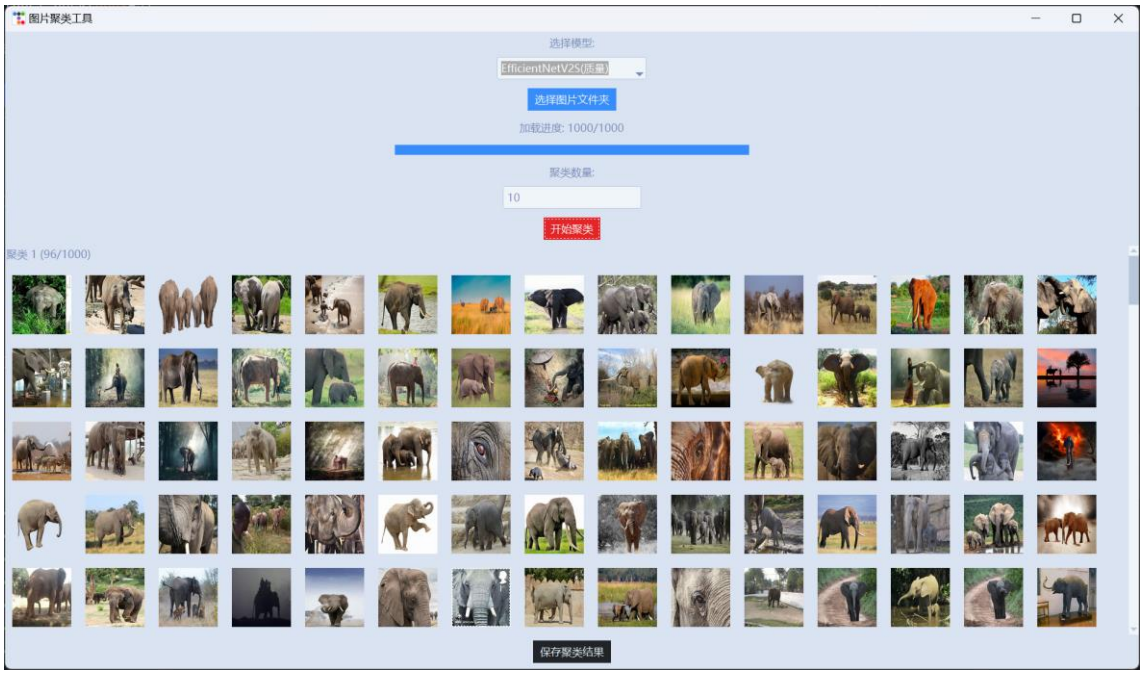


图 4-5-4 EfficientNetV2S 模型测试

测试结果如下表所示：

模型	平均加载速度（ms/张）	正确样本数	准确率
MobileNetV3Small（效率）	约 47	755	75.5%
EfficientNetB0（推荐）	约 93	837	83.7%
EfficientNetV2S（质量）	约 288	951	95.1%

表 4-5 1000 张 10 类中区分度图片测试结果

4.6 小结

该图片聚类工具成功实现了图形用户界面（GUI）的展示，用户可以便捷地通过界面选择包含图片的文件夹进行聚类操作。在完成聚类数量的输入并执行聚类过程后，聚类结果能够准确地在页面上展示出来，同时还提供了保存聚类结果的功能，方便用户对聚类结果进行后续的分析和使用。根据所使用的预训练模型参数量的不同，该工具在每张图片的加载速度上有所提升，同时聚类的准确率也得到了提高，体现了模型参数量对聚类效果的影响。这表明工具在处理不同规模和复杂度的图片数据集时，能够根据模型的选择提供不同程度的性能和准确性，满足了用户对于高效、准确图片聚类的需求。

参考文献

[1] 郭西风.基于深度神经网络的图像聚类算法研究[D].国防科技大学,2020.DOI:10.27052/d.cnki.gzjgu.2020.000056.、

[2] 聚类分析算法——K-means 聚类 详解,https://blog.csdn.net/goTsHgo/article/details/143231544

[3] Animals-10, https://www.kaggle.com/datasets/alessiocorrado99/animals10