[一. 绪论 2](#_Toc155626221)

[二.问题分析与系统设计 2](#_Toc155626222)

[2.1问题分析 2](#_Toc155626223)

[2.2系统描述 2](#_Toc155626224)

[2.2.1 CNN图像识别系统 2](#_Toc155626225)

[2.2.2 BILSTM笔画序列识别系统 3](#_Toc155626226)

[2.2.3Web系统实现 3](#_Toc155626227)

[三.数据分析与统计 3](#_Toc155626228)

[3.1 图像数据集 3](#_Toc155626229)

[3.2图像笔画序列数据集 4](#_Toc155626230)

[四.模型设计 5](#_Toc155626231)

[4.1CNN模型设计 5](#_Toc155626232)

[4.1.1CNN模型架构陈述 5](#_Toc155626233)

[4.1.2ResNet18模型架构陈述 6](#_Toc155626234)

[4.1.3CNN模型与Resnet18训练细节 7](#_Toc155626235)

[4.2BILSTM模型设计 8](#_Toc155626236)

[4.2.1BILSTM模型架构陈述 8](#_Toc155626237)

[4.2.2BILSTM模型训练细节 9](#_Toc155626238)

[五.Web系统实现细节 9](#_Toc155626239)

[六．预测模型实验测试 9](#_Toc155626240)

[6.1各CNN训练过程 9](#_Toc155626241)

[6.2CNN模型预测结果展示 11](#_Toc155626242)

[6.3BILSTM训练过程 16](#_Toc155626243)

[6.4BILSTM模型预测结果展示 16](#_Toc155626244)

[七．Web系统测试 18](#_Toc155626245)

[八.个人总结与体会 18](#_Toc155626246)

## 绪论

随着机器学习和深度学习技术的快速发展，图像识别领域取得了令人瞩目的成就。然而，大多数研究工作都集中在自然图像的识别上，对于简笔画的自动识别研究相对较少。简笔画作为一种简化的绘画形式，广泛应用于儿童教育、艺术创作和设计领域，具有重要的实际应用价值。

本次大作业主要锻炼了我们对于CNN、LSTM神经网络的应用实践能力，同时也加强了我们搭建Web应用的能力。让我们加深了对深度学习的理解，加强了我们的整体业务能力。

## 二.问题分析与系统设计

### 2.1问题分析

本次实验中主要有三大任务：

1. 构建完整的CNN图像识别系统
2. 构建完整的BILSTM 笔画序列识别系统
3. Web搭建整合CNN图像识别模型系统与BILSTM笔画序列识别系统

以及展示三大系统的效果以及数据集

针对CNN识别图像识别系统，我们首先要了解什么是CNN，本次数据集的结构是什么样的，如何选择合适的CNN网络结构适应任务的数据集，并设计合适的方式展示图像样本以及输出模型输出的概率分布。

针对BILSTM笔画序列识别系统，我们需要了解LSTM，以及什么是BILSTM,同时由于训练LSTM模型时一个批次内输入序列的长度需要一致，我们需要提前了解数据集序列的结构，针对数据集结构做出相应的数据预处理（对输入序列进行填充或者截断），根据数据集设计合适的BILSTM结构，并设计合适的方式展示图像样本以及输出模型输出的概率分布。

针对Web搭建，我们需要了解学习前后端的搭建与连接，需要实现实验数据的展示、在web界面的图像绘制与保存、使用模型预测这三种主要功能。这其中还需要解决django框架连接数据库、数据迁移、前后端跨域发送消息、使用canvas实现画图并保存功能等一系列难题。

### 2.2系统描述

#### 2.2.1 CNN图像识别系统

CNN（卷积神经网络）是深度学习中的一个基础模型，广泛用于图像识别。在我们的系统中，CNN用于处理简笔画的自动识别任务。主要步骤包括：

1. **数据预处理**：将简笔画图像转换成适合CNN处理的格式。这可能包括调整图像大小、归一化像素值等。
2. **模型设计**：选择合适的CNN架构，根据数据集的特性进行调整。例如，层数、卷积核大小、激活函数的选择等。
3. **训练与验证**：使用训练集数据训练CNN模型，并在验证集上测试其性能，以评估模型的准确性和泛化能力。
4. **结果展示**：展示模型对新输入的简笔画图像的识别结果，通常包括分类的概率分布。

#### 2.2.2 BILSTM笔画序列识别系统

BILSTM（双向长短期记忆网络）是LSTM的一种变体，它通过从两个方向处理序列数据来提高模型对上下文的理解。在笔画序列识别任务中，BILSTM的应用如下：

1. **数据预处理**：由于BILSTM处理的是序列数据，因此我们需要对简笔画的笔画序列进行预处理，包括序列填充或截断，以保证训练批次中的序列长度一致。
2. **模型设计**：设计BILSTM模型结构，包括层数、隐藏单元数等。此外，还需考虑如何将序列数据有效地输入到BILSTM中。
3. **训练与验证**：使用处理后的序列数据训练BILSTM模型，并在验证集上测试其性能。
4. **结果展示**：将模型对新输入序列的识别结果以直观的方式展示出来，例如，显示模型预测的笔画序列及其相应的概率分布。

#### 2.2.3Web系统实现

本次web系统我们采用django框架搭建后端，vue3框架实现前端，并实现前后端连接。

后端采用django框架，因为django是一个基于Python的高级Web框架，可以快速开发安全和可维护的网站，并且对于熟悉python的我们十分友好。为我们提供了以下优势：

1. 数据迁移：Django可以自动根据模型的定义生成数据库表，以及对表结构的修改，无需手动编写SQL语句。这样可以方便地同步模型和数据库的状态，避免出现不一致的情况12。
2. 连接mysql数据库方便：Django支持多种数据库后端，包括mysql，只需要在settings.py文件中配置数据库的相关参数，就可以使用Django的对象关系映射器（ORM）来操作数据库，无需关心底层的SQL细节。本次我们实现连接数据库使前端页面更灵活。

前端使用vue3框架的优点如下：

1. 更快的渲染速度：vue3通过使用Proxy代理对象和优化虚拟DOM算法等方式，提高了渲染性能。
2. 更小的包大小：vue3按需编译，只打包需要的功能模块，减少了不必要的代码。这样可以降低应用的加载时间和网络传输成本。
3. 更好的开发体验：vue3引入了新的API，如Composition API，提供了更灵活的组织代码的方式，方便了复杂组件的开发和复用。

所实现的web系统主要有三大功能模块：

1. **简笔画图像的查询及展示：**用户能够通过选择种类标签查看不同类别下的简笔画图像和简笔画序列图像。
2. **绘画模块：**用户能在指定区域内进行图画绘制，并且能够将绘制的图像保存至本地。
3. **模型预测模块：**用户能够上传指定的图片，并点击预测按钮进行预测，模型会给出在10类上的预测概率分布柱状图并与用户所选图片一同展示到页面。

## 三.数据分析与统计

本次实验提供了2个数据集，第一个数据集存储了10种类别的图像，第二个数据集存储了十种类别的图像笔画序列。两个数据集都以.npy格式存储。因此需要使用numpy库函数中的numpy.load()函数进行读取。

### 3.1 图像数据集

图像数据集中包含了10个类别的图像，每个类别存储在数据集目录下的xxx文件夹内的xxx.npy文件中，每个文件夹内包含了一张该类图片的28\*28的预览图片。首先我们通过numpy.load()读取了apple.npy与bear.npy两个文件，并打印了其形状，我们发现两个数据集的形状都是（20000，784）。随后打印了其它类别，发现类别的数据形状均一致。由此我们得知数据集内每个类别包含了20000个样本，每个样本都以一个1\*784的行向量的形式存储，数据集内共计200000个样本。

为了方便后续标签映射，我们将标签'ambulance','apple','bear','bicycle','bird','bus','cat','foot','owl','pig'映射到了0-9。我们选择在每个样本的最后一列加上对应的映射标签，并将文件存储到csv中。



图1.图像数据集具体类别



图2.npy文件内数据形状

当我们模型需要输入时，我们只需要读取0-783列数据并变换形状为28\*28作为输入，784列作为标签，就可以将多个样本组成批次送人模型进行训练。

### 3.2图像笔画序列数据集

图像笔画序列数据集中包含了10个类别的图像笔画序列数据。每个类别的数据均保存在xxx.npy文件中，通过使用numpy.load()函数读取各个类别的数据并打印各类别数据形状，我们发现各类形状为（70000，seqlen，5）,每个类别的样本数一致，序列的维度一致，但序列长度不一致。由实验提示我们得知列中每一个点用1个5维向量描 述(∆𝑥, ∆𝑦, 𝑝1, 𝑝2, 𝑝3), 𝑝1：表示画笔此时正在画图, 𝑝2：表示画笔此时离开画布 ,𝑝3：表示此时结束绘画。通过统计其形状，我们获得最长的序列长度为151。因此在后续的任务中，我们通过torch的pad函数，将所有序列填充成一致形状(70000,151,5)。



图3.图像笔画序列数据集具体类别

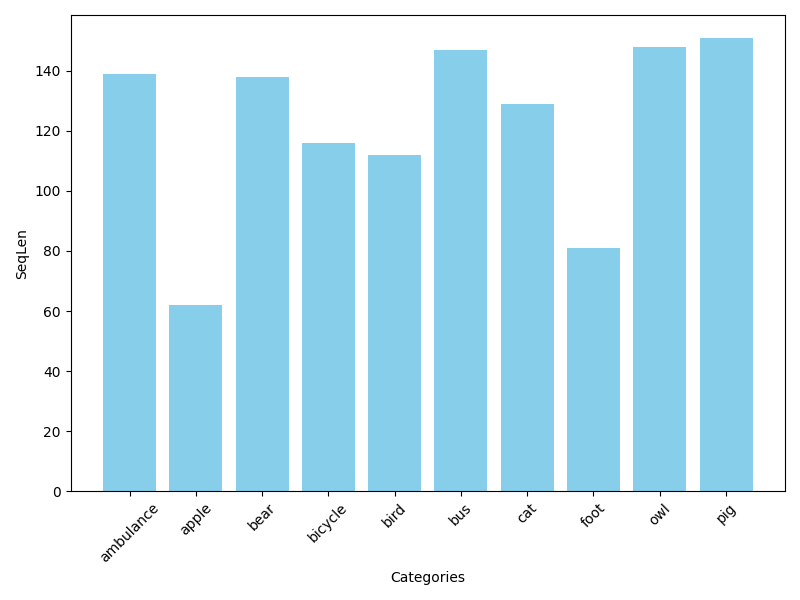


图4.各类笔画序列长度分布情况

## 四.模型设计

### 4.1CNN模型设计

本次图像识别任务中，我们使用了自搭的CNN模型，以及模仿Resnet18结构手动搭建Resnet18，并使用torch库中自带的Resnet50。

#### 4.1.1CNN模型架构陈述

本模型的搭建基于pytorch。基于数据集的结构，我们搭建了三个卷积三个卷积层和两个全连接层的CNN模型，具体结构如下：

* 第一个卷积层：输入通道数为1，输出通道数为6，卷积核大小为5x5，步长为1，无填充，使用ReLU激活函数，然后使用2x2的最大池化层进行下采样，输出特征图的大小为12x12x6。
* 第二个卷积层：输入通道数为6，输出通道数为10，卷积核大小为5x5，步长为1，无填充，使用ReLU激活函数，然后使用2x2的最大池化层进行下采样，输出特征图的大小为4x4x10。
* 第三个卷积层：输入通道数为10，输出通道数为16，卷积核大小为3x3，步长为1，无填充，使用ReLU激活函数，输出特征图的大小为2x2x16。
* 第一个全连接层：输入维度为2x2x16=64，输出维度为32，使用ReLU激活函数。
* 第二个全连接层：输入维度为32，输出维度为10，使用Softmax激活函数，得到每个类别的概率分布。

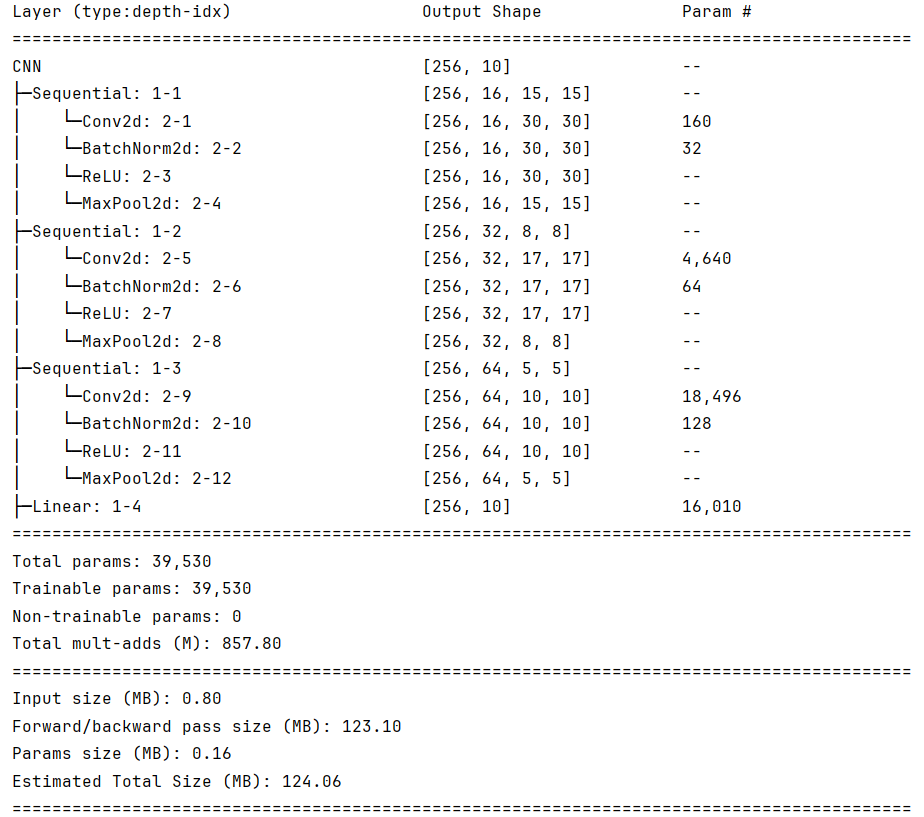


图5.CNN模型具体架构

#### 4.1.2ResNet18模型架构陈述

基于PyTorch框架，我们构建了一个专门针对简笔画识别的ResNet18模型。该模型的结构如下：

1. **修改的第一层**：为了适应简笔画的单通道输入，我们将第一层卷积层的输入通道数修改为1，输出通道数为64，卷积核大小为7x7，步长为2，带有3个像素的填充，无偏置项。接着是批量归一化和ReLU激活函数，以及最大池化层（核大小为3x3，步长为2，填充为1）。
2. **残差块层**：模型包含4个残差块层，每个层级有不同数量的残差块。每个残差块包含两个卷积层，其中每个卷积层的核大小为3x3，带有1个像素的填充，和批量归一化。如果需要，会在残差块中添加下采样层来调整维度。
   * 第一层（layer1）: 64个输出通道，包含layers[0]个残差块。
   * 第二层（layer2）: 128个输出通道，包含layers[1]个残差块，步长为2。
   * 第三层（layer3）: 256个输出通道，包含layers[2]个残差块，步长为2。
   * 第四层（layer4）: 512个输出通道，包含layers[3]个残差块，步长为2。
3. **自适应平均池化**：该层将特征图的大小调整为1x1，为全连接层提供输入。
4. **全连接层**：最后一层是一个全连接层，输出维度为类别数（num\_classes）。

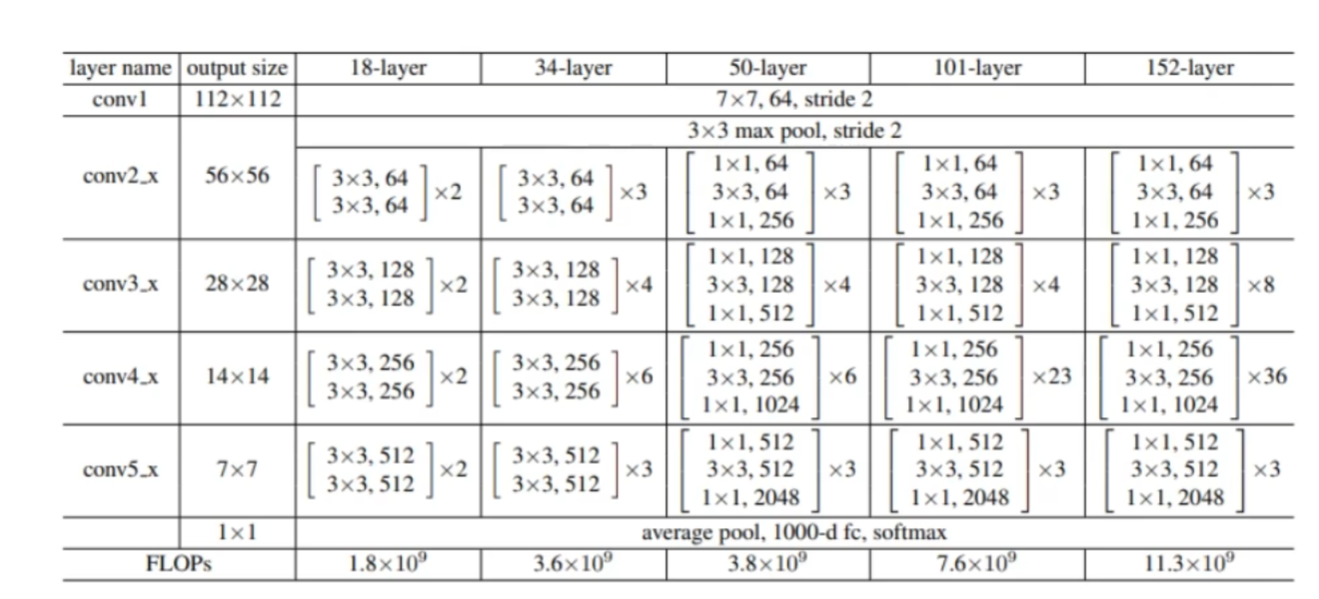


图6.各Resnet结构图

#### 4.1.3CNN模型与Resnet18训练细节

本节详细介绍了基于PyTorch的CNN模型与ResNet18模型在简笔画识别任务中的训练过程及相关细节。

1. **模型结构**：
   * **CNN模型**：定义了三个卷积层，每层均包含卷积、批量归一化、ReLU激活函数和最大池化。卷积层的输出通道数分别为16、32和64。最后接一个全连接层，用于分类。
   * **ResNet18模型**：基于残差块构建，每个残差块包含两个卷积层。整个网络包括4个残差块层，每个层包含2个残差块。初始卷积层被修改为单通道输入，以适应28x28的简笔画。
2. **数据处理**：
   * 数据从CSV文件中加载，并转换为PyTorch张量。
   * 图像数据假设为28x28的大小，转换为对应的张量格式。
   * 使用train\_test\_split函数将数据集分为80%的训练集和20%的测试集。
3. **训练过程**：
   * 使用CrossEntropyLoss作为损失函数，适用于多分类问题。
   * 选择Adam优化器，初始学习率设置为0.003。
   * 设置批处理大小为256，考虑到数据集和模型的具体情况。
   * 训练过程中，每完成一定数量的批次，打印当前损失值。
   * 使用GPU加速训练（如果可用）。
4. **模型评估与保存**：
   * 在测试集上评估模型的准确率。
   * 保留表现最佳的模型参数，保存到指定路径。

通过以上细节的配置和优化，CNN模型和ResNet18模型能够高效且准确地进行训练，从而在简笔画识别任务中实现良好的性能

### 4.2BILSTM模型设计

#### 4.2.1BILSTM模型架构陈述

本模型的搭建基于 pytorch。基于数据集的结构，我们搭建了一个双向长短期记忆网络（BiLSTM）和一个全连接层的分类器模型，具体结构如下：

* 第一个 BiLSTM 层：输入维度为 5，输出维度为 128，层数为 5，使用 dropout 概率为 0.5，双向为 True，将批次维度放在第一位，使用 tanh 激活函数，输出一个双向的隐层状态序列，大小为 512x5x256（批次大小x序列长度x隐层维度x2）。
* 第一个全连接层：输入维度为 256，输出维度为 10，将双向的隐层状态序列的最后一个时刻的输出作为输入，使用 Softmax 激活函数，得到每个类别的概率分布。

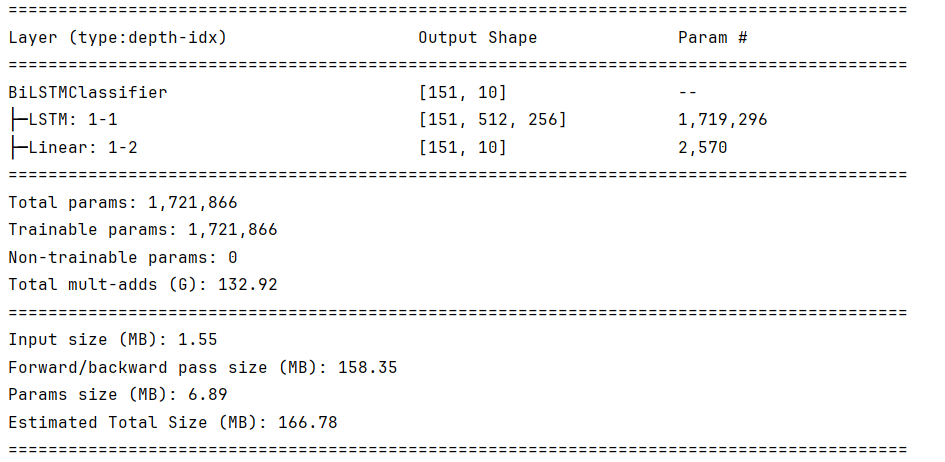


图7. BILSTM网络结构

#### 4.2.2BILSTM模型训练细节

在BILSTM模型的训练过程中，我们采取了一系列细致的步骤和参数配置，以确保模型能够有效地学习并识别简笔画的序列特征：

1. **数据集处理**：首先，我们使用了一个自定义的SketchDataset类来处理和加载数据。该类读取不同类别的简笔画数据，并将每个样本转换为统一长度的序列，以适应BILSTM模型的输入要求。这是通过对较短的序列进行填充来实现的。
2. **模型结构**：我们构建了一个BiLSTMClassifier模型，该模型包含一个双向LSTM层和一个全连接层。双向LSTM层有助于模型从序列的两个方向捕捉上下文信息，而全连接层用于最终的分类。
3. **优化器与损失函数**：我们使用了Adam优化器，具有0.001的学习率，以及交叉熵损失函数，这在处理多分类问题时非常有效。
4. **训练策略**：训练过程中，我们设置了大批量大小（batch\_size=512），以提高内存利用率和训练效率。训练了10个周期（epochs），并使用了早期停止策略来防止过拟合。
5. **评估和保存模型**：在每个训练周期后，我们使用测试集评估模型的准确度，并将准确率记录到日志文件中。训练完成后，我们将模型的参数保存到文件中，以便于后续的使用和进一步的分析。
6. **设备选择**：根据硬件条件，我们选择在GPU上训练模型（如果可用），以加速训练过程。

通过这些细节的配置和调整，BILSTM模型能够更准确地识别和处理简笔画的序列数据，从而提高了整体的识别效率和准确性。

## 五.Web系统实现细节

### 5.1 django框架与vue3框架相关配置

drf-REST framework是一个基于Django框架的Web API开发框架，它提供了一些优势：

* 1. 缩减了编写API接口的代码，可以使用序列化器对象Serializer的定义，来简化序列化与反序列化的过程，无需手动处理JSON或其他格式的数据34。
  2. 提供了丰富的类视图、扩展类、视图集来简化视图的编写工作，可以根据不同的请求方法，自动调用相应的处理函数，无需重复编写代码34。
  3. 支持多种身份验证、分页、过滤、搜索、排序等功能，可以通过配置或自定义来实现API的安全性和灵活性。

配置drf-REST framework，主要是为了利用drf-REST framework的优势，来快速、高效、规范地开发RESTful风格的API。

RESTful风格的API是一种遵循REST原则的Web服务接口，它可以满足前后端分离的架构设计需求，主要有以下几个方面：

* + 1. 前后端职责清晰：RESTful API将前端的展示层和后端的业务逻辑层分离，使得前后端可以独立开发、测试和部署，降低了耦合度，提高了开发效率和可维护性123。
    2. 接口标准化：RESTful API使用统一的HTTP方法和URI来操作资源，遵循一致的命名规范和数据格式，使得接口易于理解和使用，方便了前后端的协作和沟通1245。
    3. 数据传输高效：RESTful API支持多种数据格式，如JSON、XML等，可以根据不同的客户端需求，灵活地选择合适的数据格式，减少了数据的冗余和传输量，提高了网络性能1245。
    4. 可扩展性强：RESTful API可以方便地添加、修改或删除资源，无需改变现有的接口，也可以支持多种客户端，如Web、移动、桌面等，适应不同的场景和需求。

为此我们需要安装对应的djangorestframework包并修改setting.py中的相关配置。完成上述配置运行manage.py便能得到如下效果：

表格

描述已自动生成还可以注册超级用户在django自动生成的admin后台对django模型及数据库中的数据进行创建、更新、删除等操作。效果如下：

图 8 Django Rest framework效果图

图形用户界面, 文本

描述已自动生成

图 9 django的admin后台

对于前后端需要进行跨域数据传输，所以要在前后端都进行跨域设置。

文本

描述已自动生成在django后端，需要下载对应的包：django-cors-headers，并在setting.py文件中设置MIDDLEWARE的配置，并在最后加上一些配置：

在vue3框架中需要下载axios包并在main.js中引用

图 10 跨域设置配置

要实现前后端传输图片类型等数据。还需要实现django图片类型访问，设置vue图片渲染。

文本

描述已自动生成即需要在django的setting.py中设定上传的地址：

图 11 设置图片上传路径

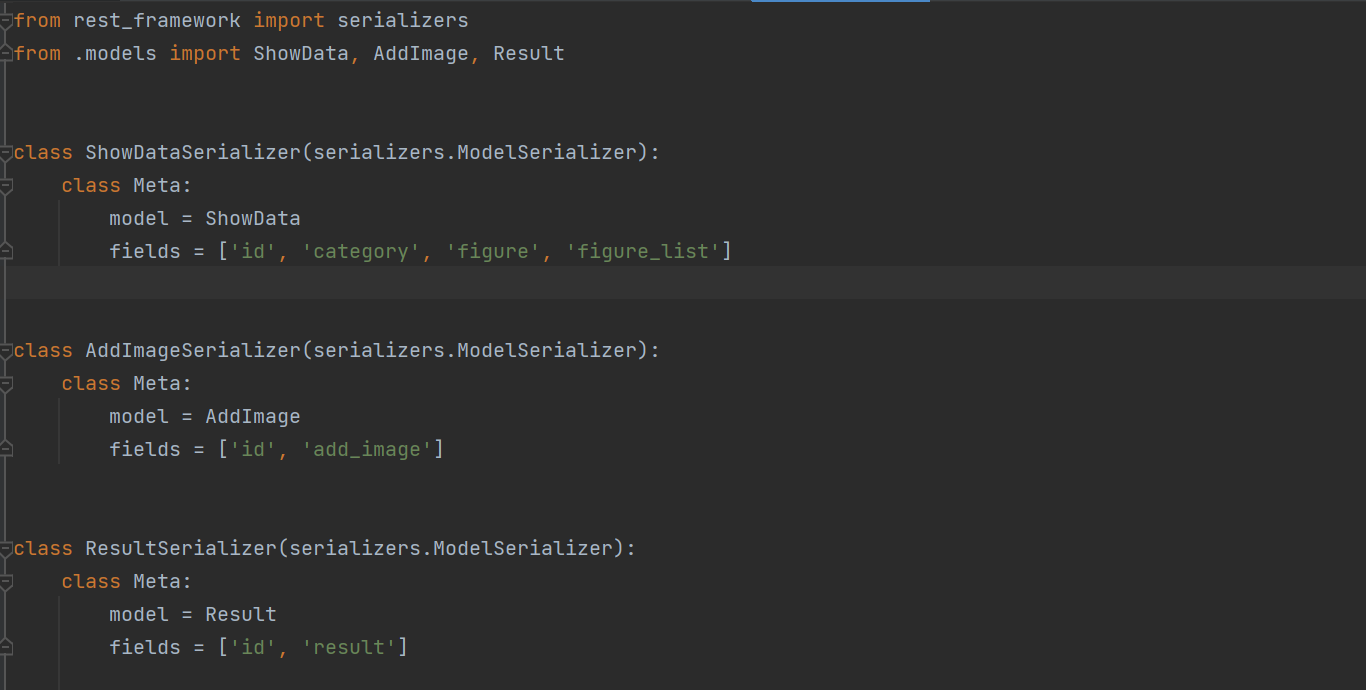
本节对django框架及vue3框架的相关配置主要为了之后设计web系统对应功能模块的方便快捷，admin超级用户的注册使设计web系统的人员直观方便的管理django模型，restframework框架使之后前后端通过axios获取数据更加快捷方便，跨域设置主要解决前后端不同域名之间的信息传输，没有此设置前后端很难交互。

### 5.2 数据展示功能实现

数据展示模块功能相对简单，用户通过选择器选择想要参看的数据标签，点击按钮即可看到简笔画图像及简笔画序列图。

对于选择器，vue3的组件能轻松实现默认标签显示、点击弹出下拉页表，选择自动获得对应标签的标号（方便用户请求数据）。对于图片显示使用v-show便能实现未选择时将没有的图像隐藏。

本模块的核心在于使用axios实现前端向后端的showData接口发送请求，返回接受对应图像。要实现此功能需要首先在数据库中存入每类标签对应的图片，创建对应的showData模型，编写序列化函数并定义接口能显示的数据。得到返回的数据后，vue3便能动态显示。

图 11 设置图片上传路径

接口获取数据可视化：

当用户点击对应标签时，前端自动获取其对应的标号，再通过axios向接口请求数据，参数为标号，便能获取对应标签的图片。效果如下：表格

中度可信度描述已自动生成

图 12 showData获取数据可视化

### 5.3 绘制图像模块实现

图 12 showData获取数据可视化

绘制图像模块实现了用户在指定区域内能进行绘制，点击保存按钮，可以将绘制的图片保存至本地以供后续模型识别。在本模块中主要的问题有对于canvas模块的使用、画布大小与训练数据不一致导致需要调节各种参数方便预测时对图片的压缩，将画布中的图像如何保存至本地。

canvas模块是一个可以使用JavaScript来绘制图形的HTML元素，它可以用于绘制图表、制作图片构图或者制作简单的动画。canvas模块有以下特点：

灵活性：canvas模块可以绘制任何形状和颜色的图形，也可以对图形进行变换、裁剪、填充、描边等操作。

性能：canvas模块可以利用GPU加速，提高绘图的速度和效率。

兼容性：canvas模块可以在大多数现代浏览器中运行，也可以通过polyfill来支持旧版本的浏览器。

Canvas的使用只要在template中创建canvas元素，获取canvas对象即可，主要工作是要编写正确的JavaScript代码来达到想要实现的效果；

我们设置画布在web端的背景为黑色，画笔为白色，最终保存出的图片为透明底白色笔画的图像，但是可以通过后端处理轻松变为黑底白画的灰度图。

图片存储功能通过创建link常量捕获目标的url并设置保存的名字便能实现图片保存功能。

最终实现效果如下：

图片包含 图形用户界面

描述已自动生成值得一提的是由于我们的画布设置的是560px\*560px，在之后预测时为了预测的准确度我们将传入图片都缩放至28px\*28px的大小会导致一定程度的失真。所以要求画笔的粗细要足够大并且经过多次人为测试，发现当画笔的像素为20px时模型的预测结果较好。

图 13 绘制图像模块

文本

描述已自动生成

图 14 画布效果主要JavaScript代码

### 5.3 模型预测模块实现

模型预测模块实现用户上传图片，前端传输至后端并同时对输入的图像进行预处理转化为.npy文件，方便模型预测，用户通过预测按钮使模型预测传入图像的标签并输出对应10类标签的概率分布并与用户输入的图像一同展示。

1.图像的传输与图片预处理：

前端通过创建FormData数据结构获取用户上传的图片文件，并通过axios的get请求向已经设计好的add\_image端口发送请求，便能在后端的upimg/文件夹下保存图片。

在进行预测之前需要对图像预处理，其中需要用到包imageio、PIL、cv2中的函数。

首先对传入的图片缩放至28px\*28px，使用cv2.resize函数变能实现，但是由于将大图片缩放至小图像时会产生失真，而失真程度是由缩放策略决定的。

OpenCV提供了5中方法，如下：

cv2.INTER\_NEAREST # 最邻近差值法

cv2.INTER\_LINEAR # 双线性差值法

cv2.INTER\_AREA # 基于局部像素的重采样法

cv2.INTER\_CUBIC # 三次差值法

cv2.INTER\_LANCZOS4 # 基于8\*8像素邻域的Lanczos差值法

通过调查发现使用双线性差值法压缩时失真程度最小，故采用此方法压缩图像。

接着需要通过PIL.Image的convert方法将RGBA的PNG图像转化为L（灰度图不含透明度）的PNG图像。

最后通过imageio的imread函数和np.sae函数将图像数据转化为.npy格式。

2.模型预测与模型接口搭建:

Django后端不能直接导入包含深度学习包的预测函数，需要将预测函数封装成接口api的形式通过后台终端调用。

我们使用pywsgi.WSGIServer实现此api的封装：

文本

描述已自动生成

图 15 封装预测函数

前端通过预测按钮执行test函数，先后端发送请求，对已经处理好的传入的图片进行预测，并将结果保存至upimg/文件夹下实现将预测结果传输至前端。

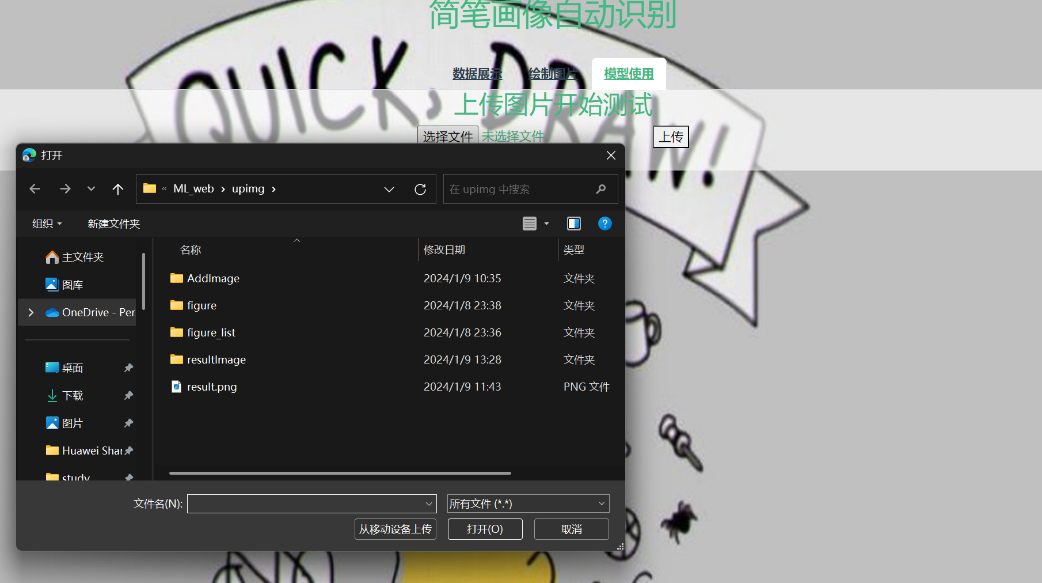
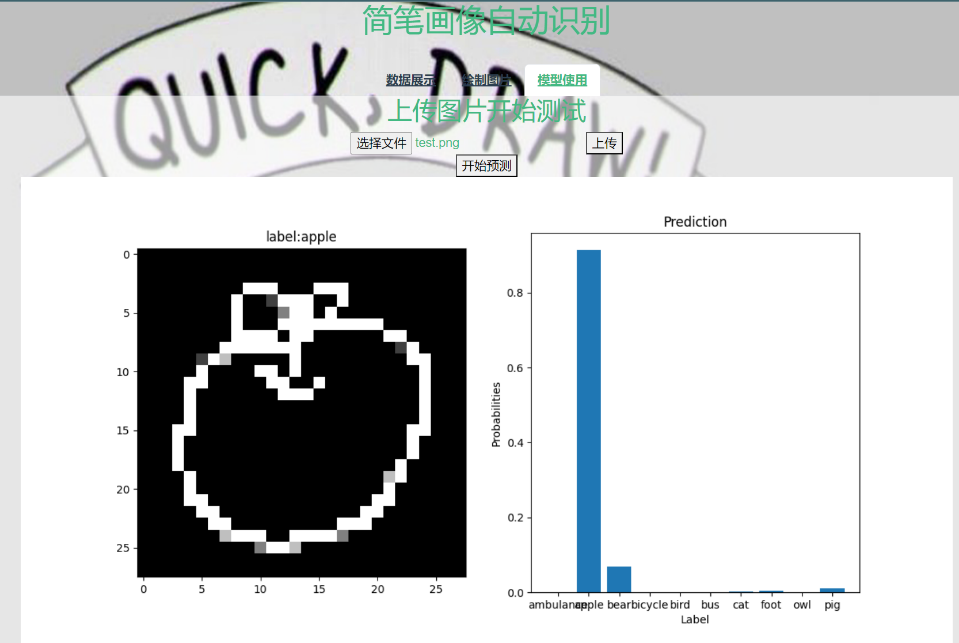
最终效果如下：

图 17 预测结果展示

图 16 图片上传功能

## 六．预测模型实验测试

### 6.1各CNN训练过程

在本研究中，我们分别对CNN、ResNet18和ResNet50模型进行了详细的训练和评估。下面是这三个模型的训练过程总结：

**CNN模型：**

* + - * 训练周期：CNN模型进行了总共20个epoch的训练。
      * 最高准确率：在验证集上达到的最高准确度为88.6%。
      * 平均准确率：整个训练过程中的平均准确率为87.7%。
      * 损失值和准确度：训练中的平均损失值及每个epoch在验证集上的 准确度记录在图8中。

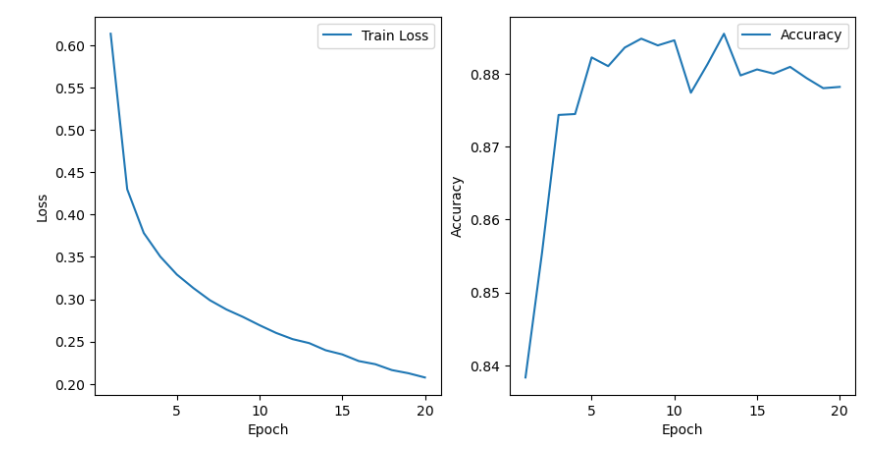


图8.CNN训练过程中的损失值与各epoch验证集精度

**ResNet18模型**：

* + 训练周期：ResNet18模型也进行了20个epoch的训练。
  + 最高准确率：在验证集上的最高准确度为89.01%。
  + 平均准确率：平均准确率为88.0%。
  + 损失值和准确度：详细的损失值与各epoch的验证集精度情况展示在图9中。

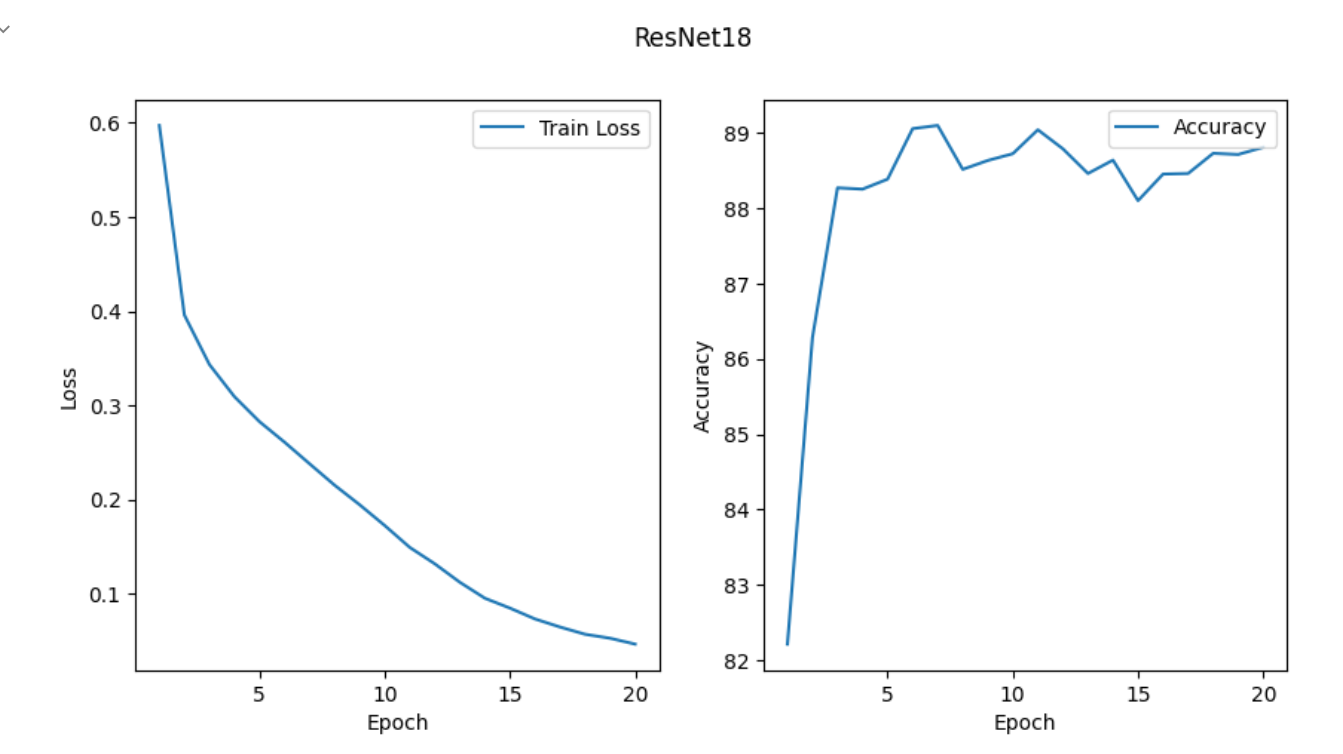


图9.ResNet18训练过程中的损失值与各epoch验证集精度

**ResNet50模型**：

* + 训练周期：ResNet50模型训练了20个epoch。
  + 最高准确率：验证集上的最高准确度同样为89.01%。
  + 平均准确率：相较于其他两个模型，平均准确率较低，为80.931%。
  + 损失值和准确度：与ResNet18模型类似，ResNet50模型的损失值和各epoch的验证集精度情况也记录在图9中。

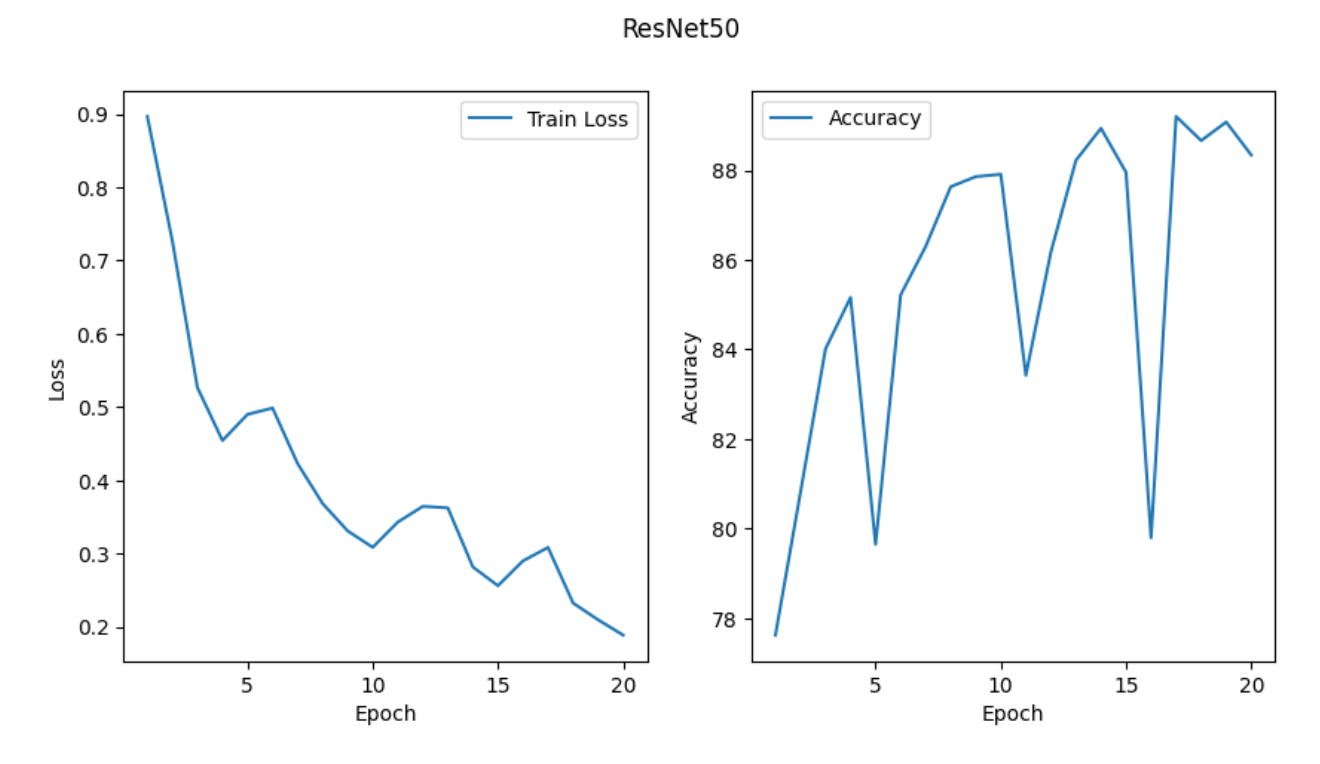


图9.ResNet18训练过程中的损失值与各epoch验证集精度

通过对这三个模型的对比，我们发现ResNet18模型在准确率和稳定性方面表现最佳。而ResNet50模型，尽管在某些epoch达到了较高的准确率，但其平均准确率较低，这可能是由于模型层数过深导致的过拟合。这些发现表明，在简笔画识别这种相对简单的任务中，过于复杂的模型可能不会带来额外的性能提升，反而可能导致效果不如预期。因此，选择合适的模型结构对于优化性能至关重要。

### 6.2CNN模型预测结果展示

从各个类的数据集中抽取10个随机样本，打印出其图片，并显示其预测类标与模型输出预测概率。由于plt输出图片原因导致显示较小，具体放大图片可到上交材料中.ipynb文件中查看。以下提供ambulance类的放大图片。



图10

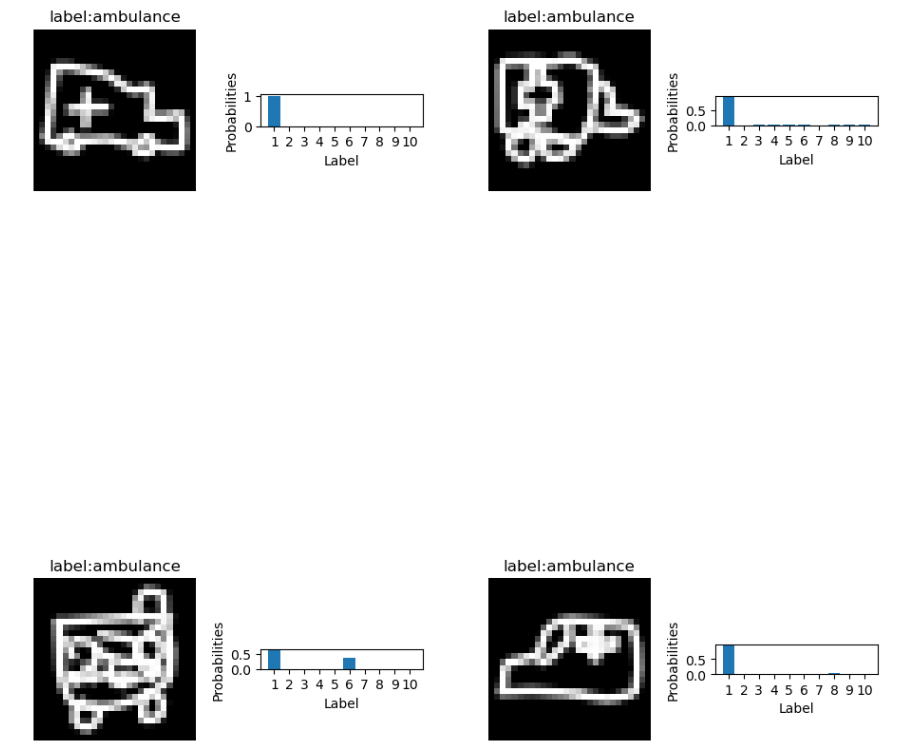
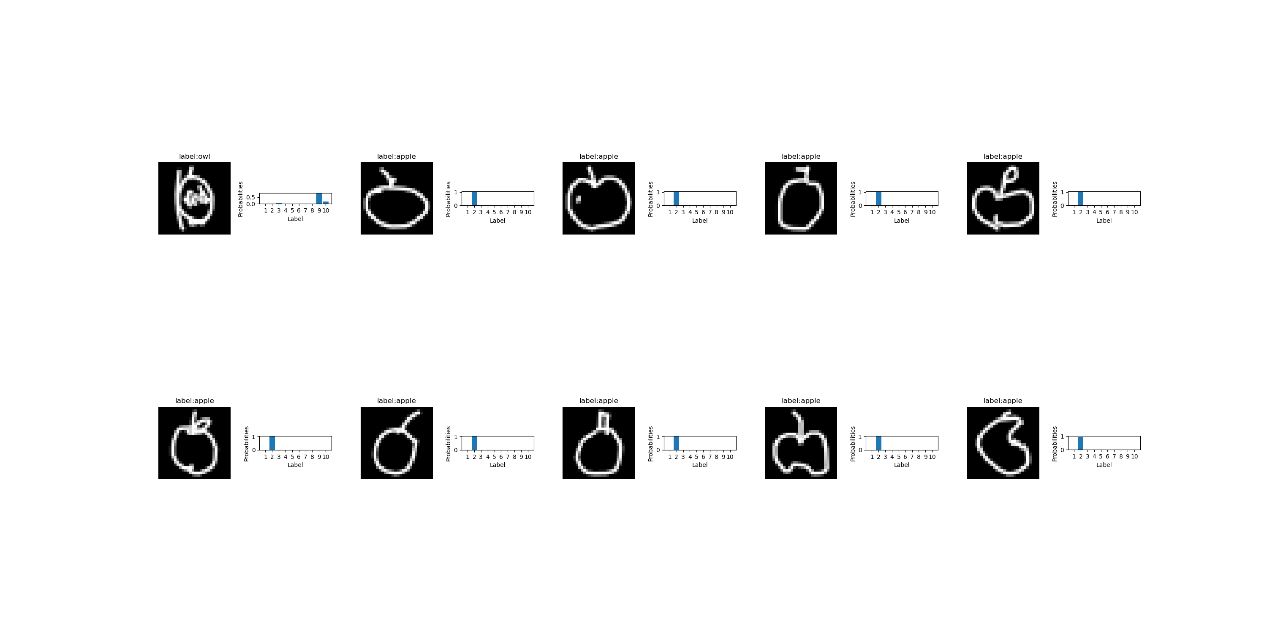
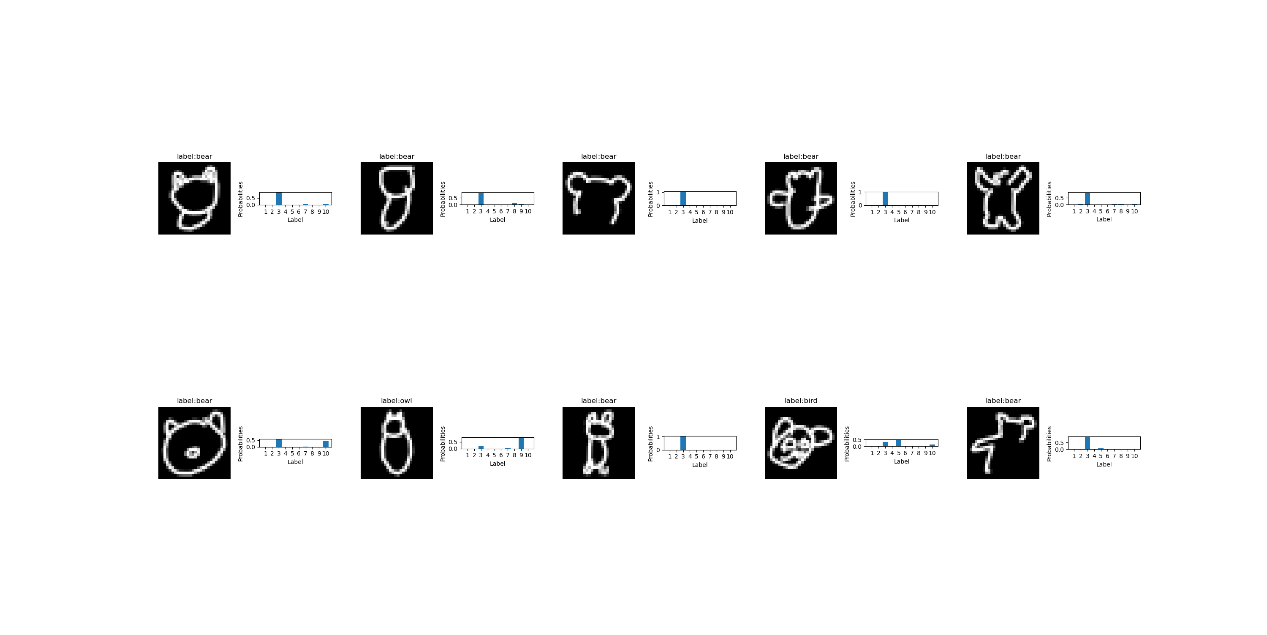
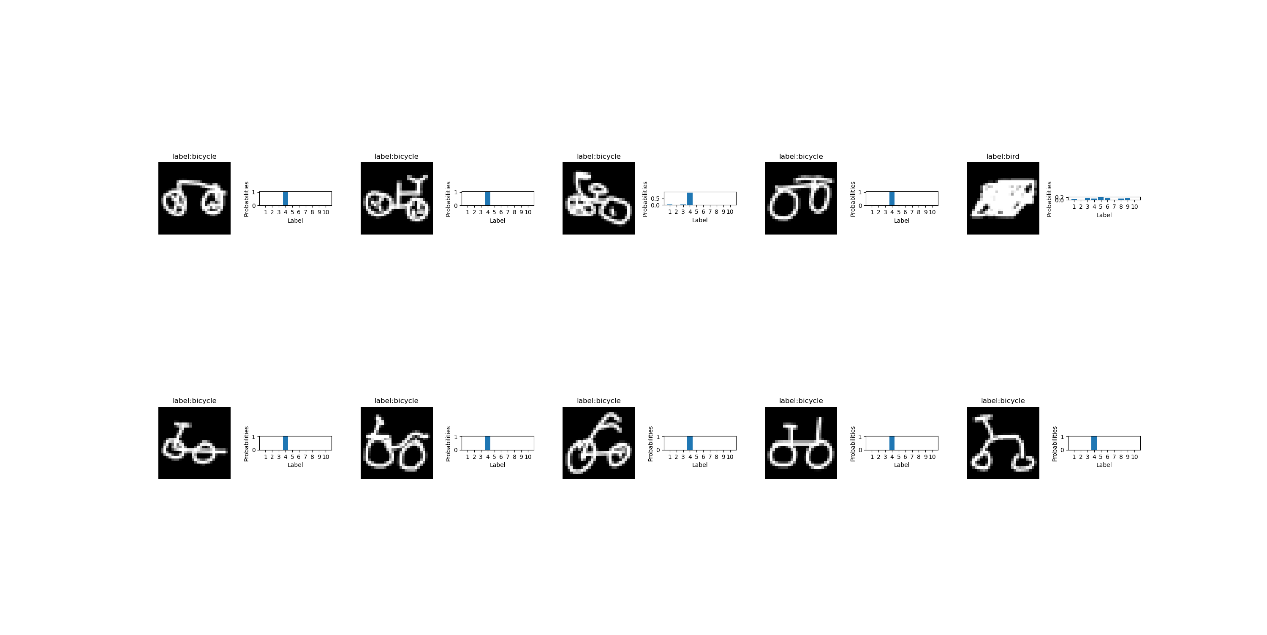
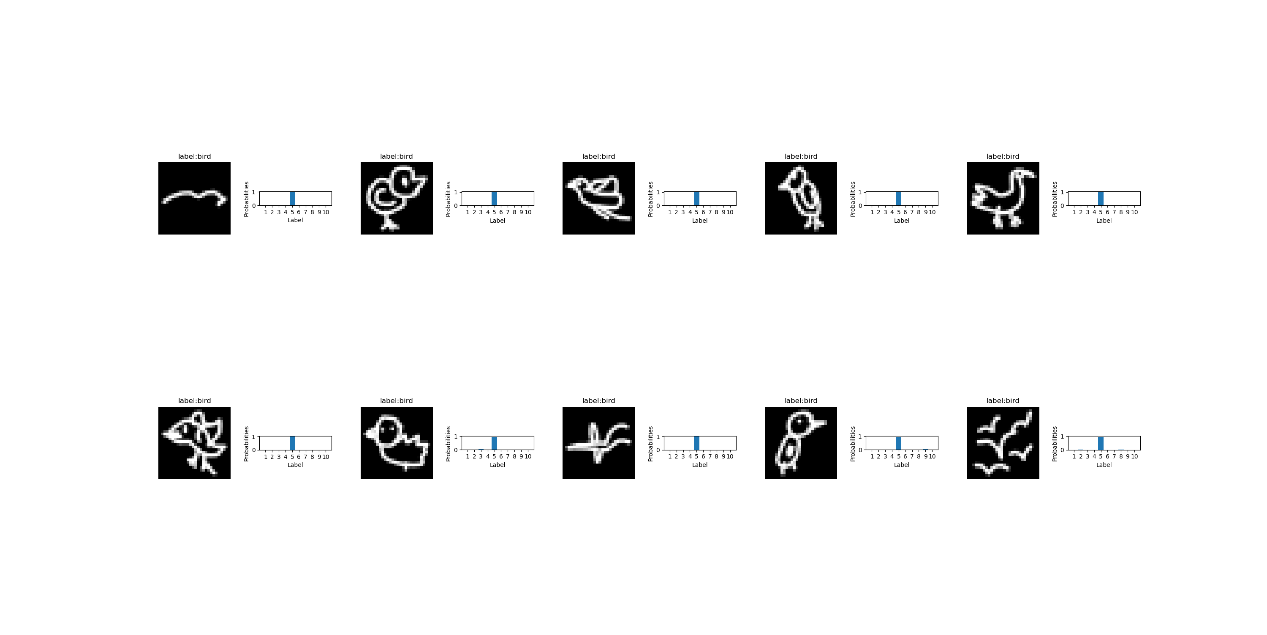


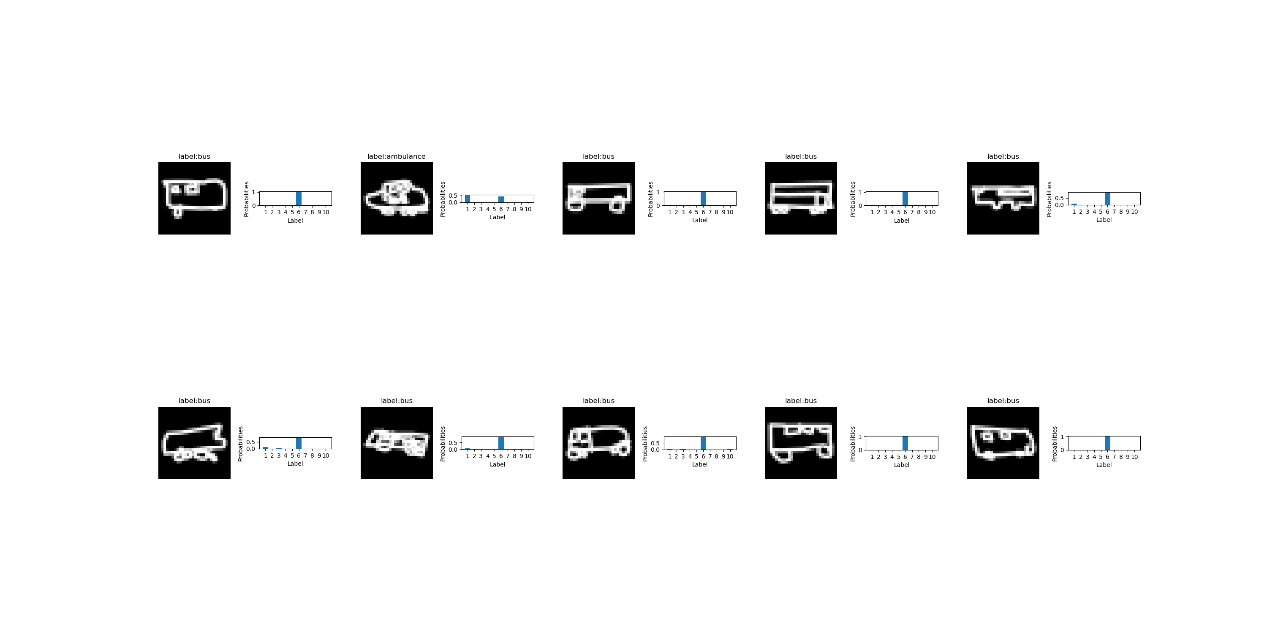
图11

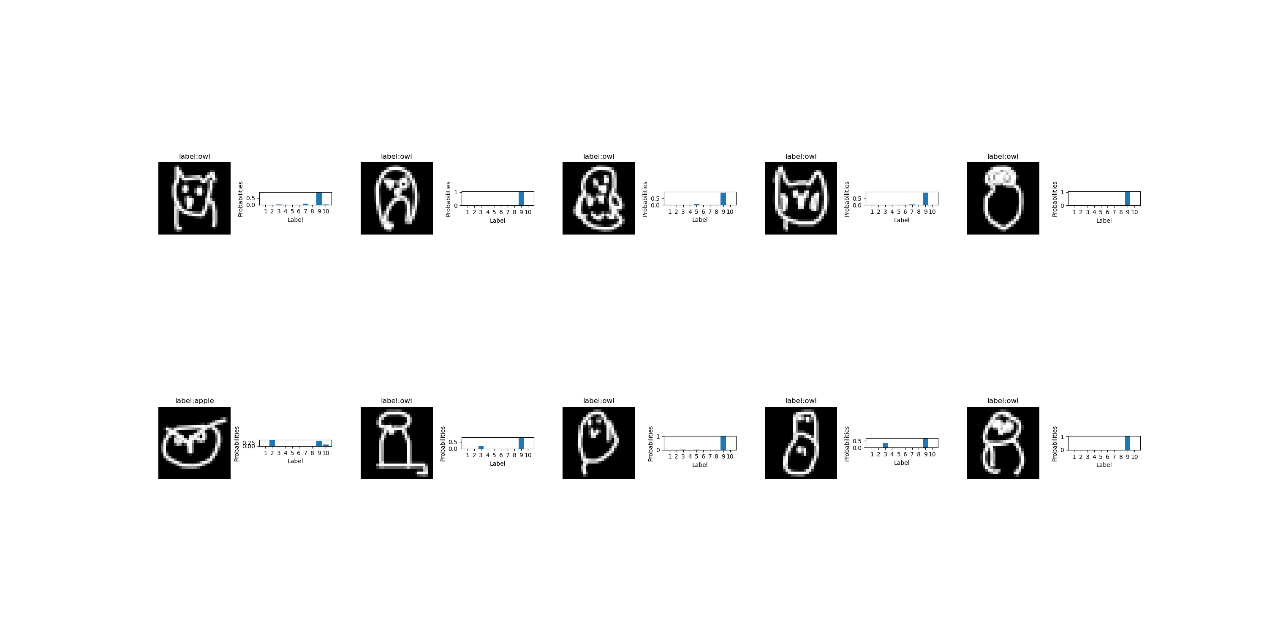
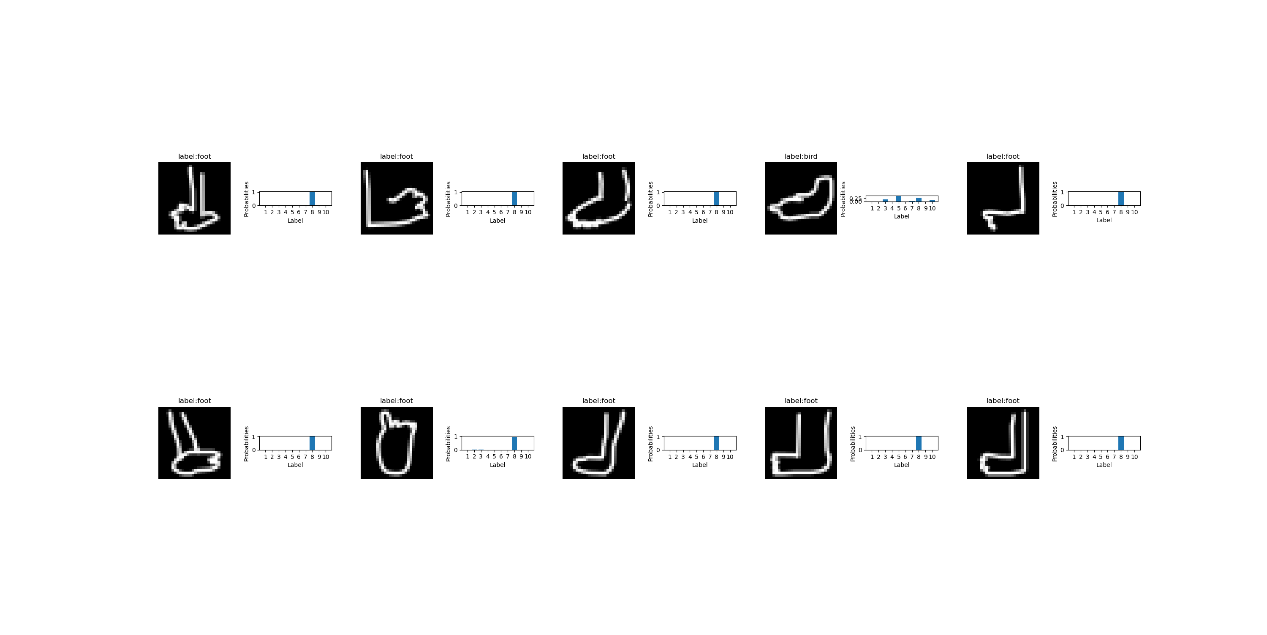












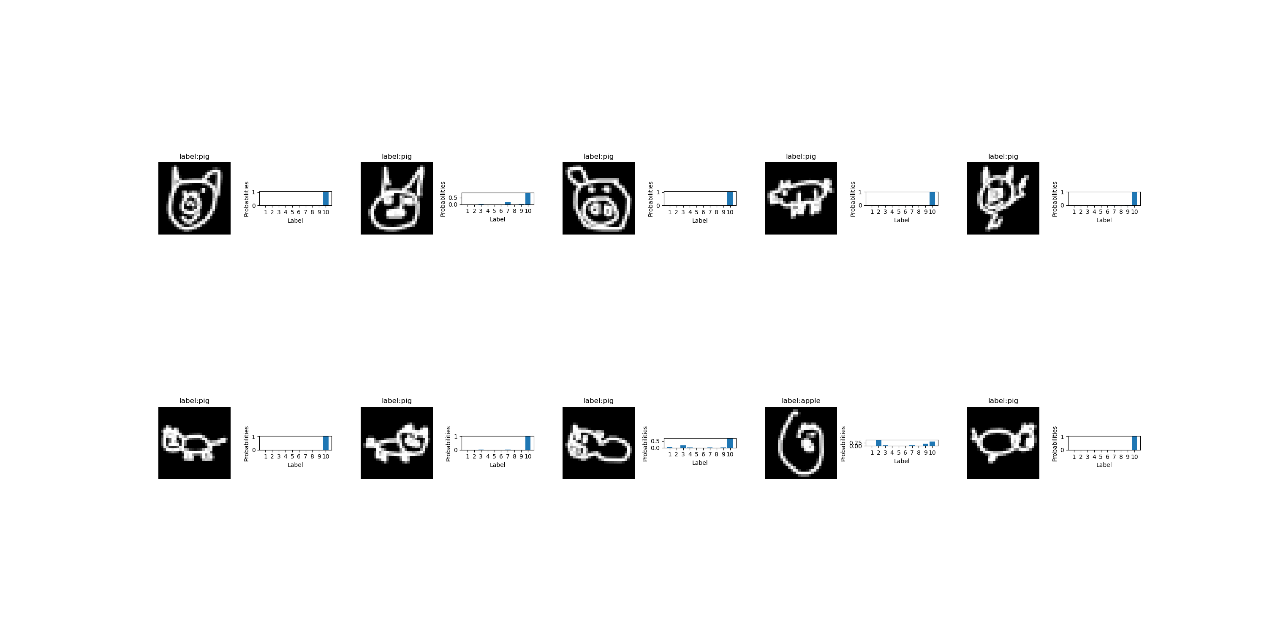


图12.9类图片随机抽样展示

### 6.3BILSTM训练过程

本模型总共训练了10个epoch，我们保存了在验证集上准确度最高的模型。在验证集最高的准确度为99.9%,平均准确度为95.5%。训练中的平均损失值，以及每个epoch的准确度如图13所示。

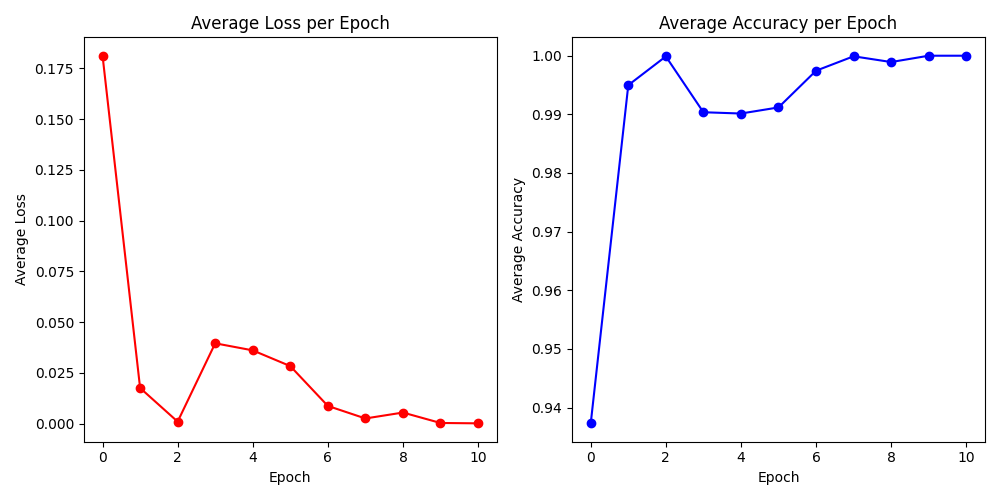
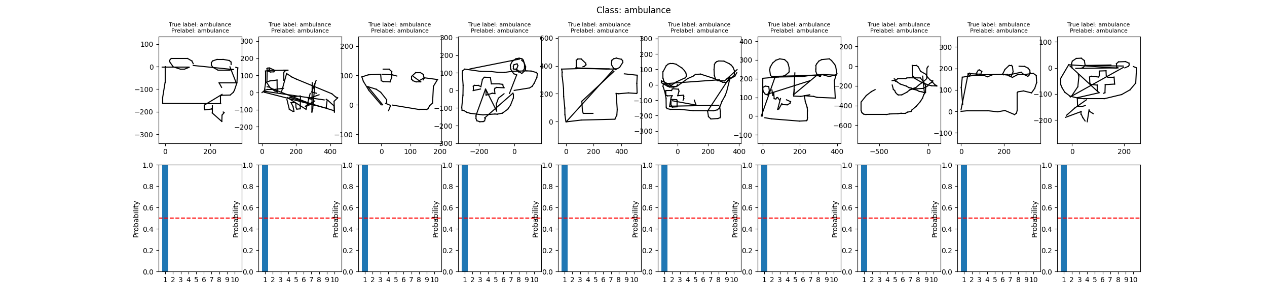
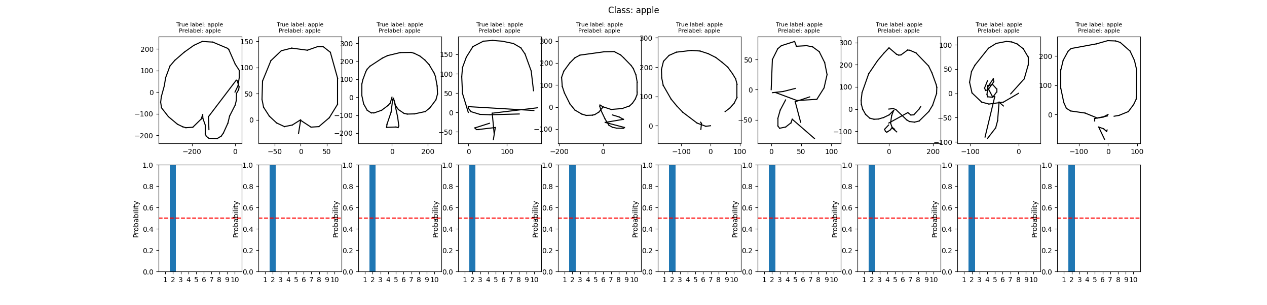


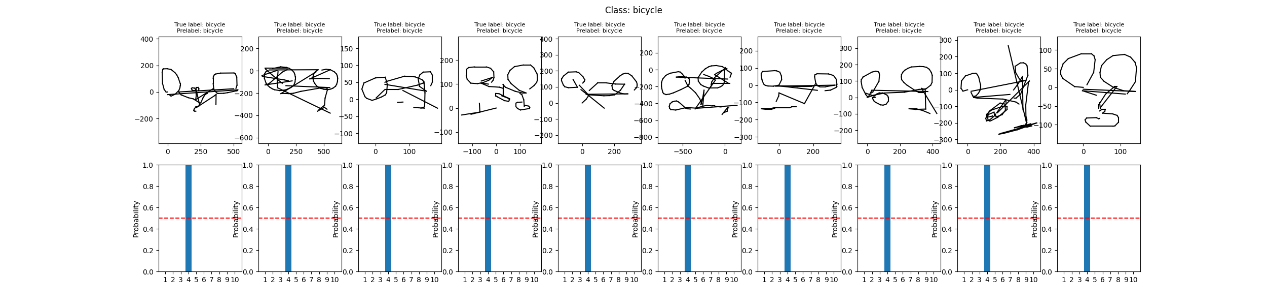
图13.loss与acc变化图

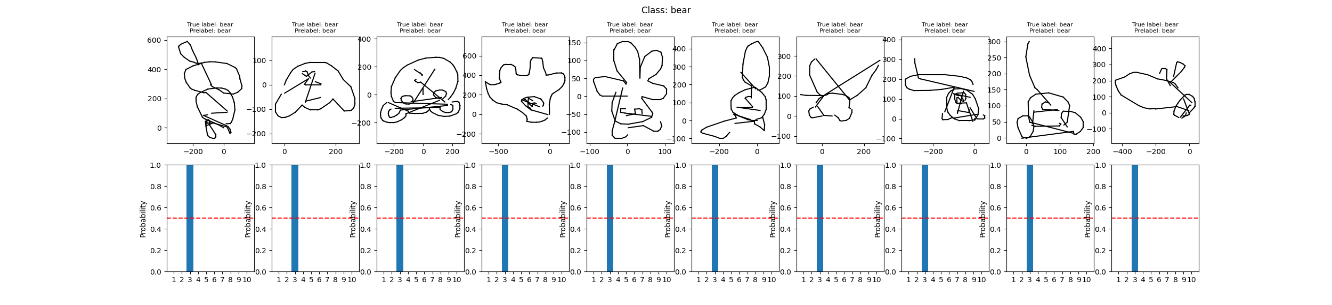
### 6.4BILSTM模型预测结果展示

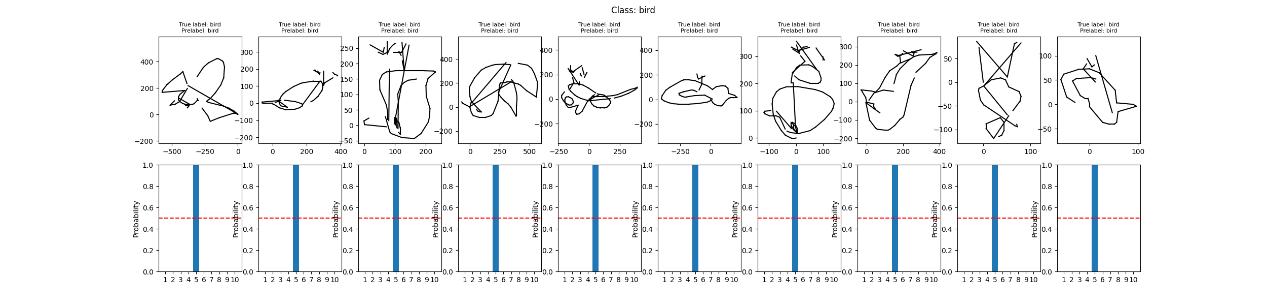
从各个类的数据集中抽取10个随机样本，打印出其图片，并显示其预测类标与模型输出预测概率。

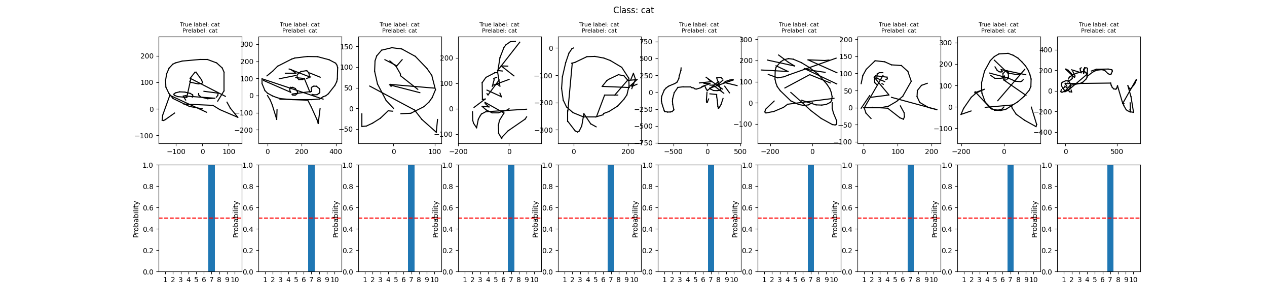


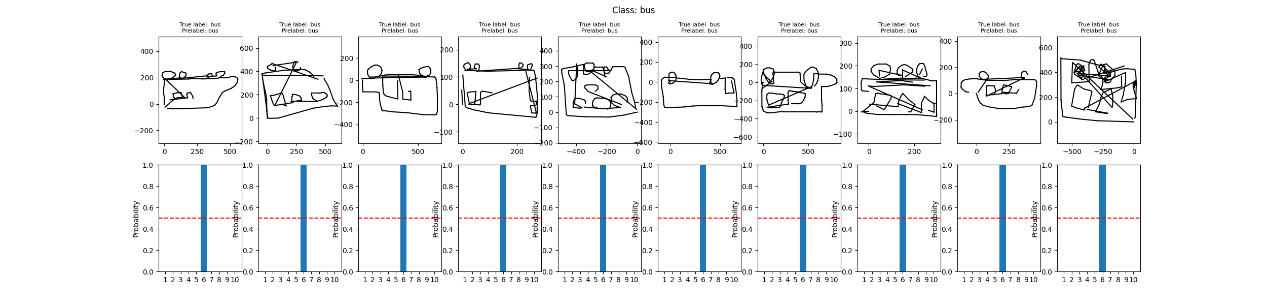


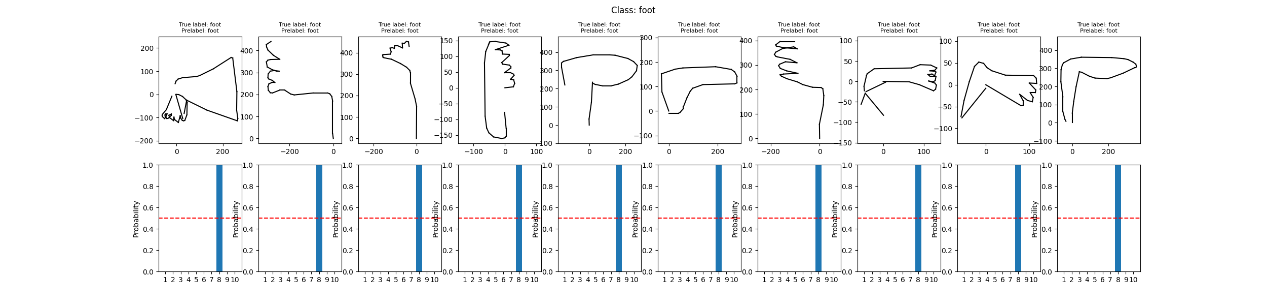


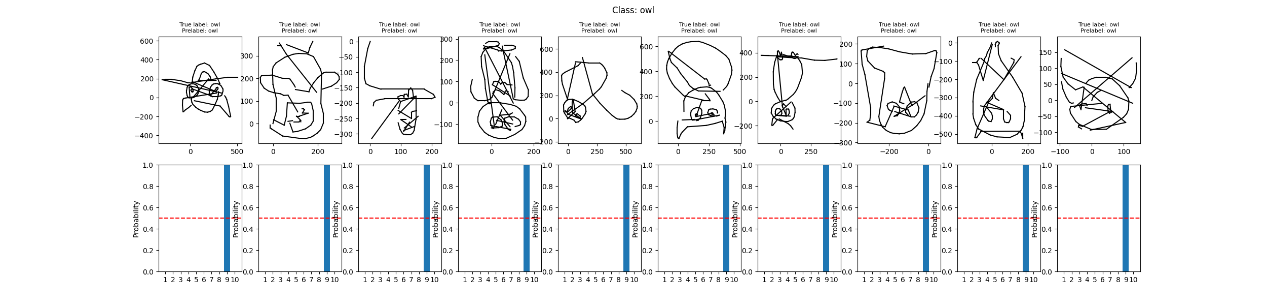












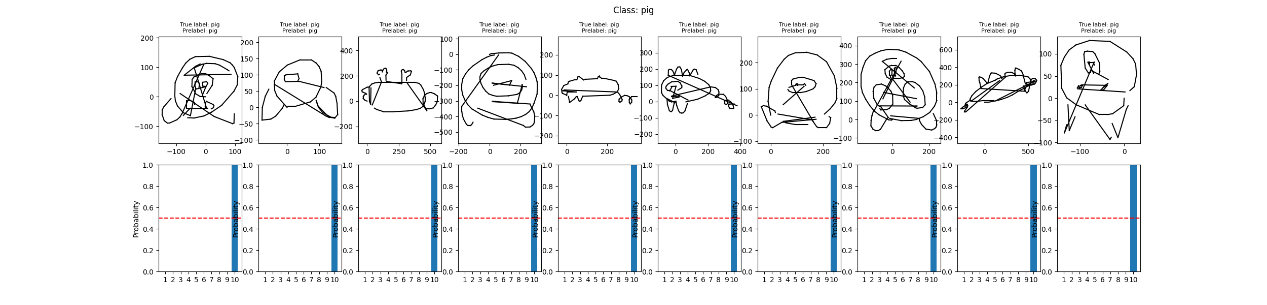


图14. 十类随机样本展示

## 七．Web系统测试

由于我们在web端的开发不够深入，对于web系统测试也掌握甚少，只进行了肤浅的系统测试，且大多数为自我评价缺乏权威性

1．UI测试

我们认为我们的UI风格相对统一，且实现了各页面的大小一致；同样的LOGO图片在各个页面中显示大小一致；页面及图片居中显示 、各页面的title正确；没有错别字或乱码；同一级别的字体、大小、颜色是统一；切换窗口大小，将窗口缩小后，页面能按比例缩小或出现滚动条；整体来看UI方面良好。

2．连接测试

页面没有无法连接的内容；图片都能正常显示，没有冗余图片且能动态显示图片文字；页面不存在死链接，但是显示的图片没有连接url，点击图片还不能进行放大显示，功能性不过完善，不够人性化；页面点击LOGO下的一级栏目或二级栏目名称，可以进入相应的栏目。整体来看连接方面一般。

3．表单提交测试

提交按钮都已实现；但是修改表单是没对重名做验证，提交数据能正常保存到后台数据库中（后台数据库中数据应与前台录完内容完全一致，数据不会丢失或被改变）表单提交，删除，修改后有提示内容。浏览器前进、后退、刷新按钮，可能会造成数据库重现；整体来看表单提交方面良好。

4．兼容性测试

由于使用vue3框架，其自身的兼容性也很高，对于浏览器兼容度较好，预计只有IE浏览器无法兼容。兼容性方面优秀。

## 八.个人总结与体会

在本次大作业中，我们围绕机器学习与深度学习技术的发展，特别关注了简笔画的自动识别问题。通过实际应用CNN和LSTM神经网络，我们不仅锻炼了技术实践能力，还增强了搭建Web应用的技能。这些经验让我们对深度学习有了更深刻的理解，并提升了整体业务能力。

我们的工作主要包括以下几个方面：

1.系统搭建与设计：我们构建了一个基于CNN的图像识别系统和一个基于BILSTM的笔画序列识别系统，并将这两个系统整合到一个Web平台中。这不仅展示了模型的效果，还提供了一个互动界面供用户体验。

2. 数据处理与模型实现：在CNN图像识别系统中，我们处理了不同类别的图像数据，并设计了一个由三个卷积层和两个全连接层组成的模型。与此同时，在BILSTM系统中，我们处理了笔画序列数据，并设计了一个结合双向长短期记忆网络和全连接层的模型。

3. 模型训练与展示：我们对这两个模型进行了训练和评估，并在Web应用中展示了模型的预测结果。特别地，我们提供了用户互动界面，允许用户图像绘制并保存图像，上传图像并查看模型的预测结果。

4. Web应用开发：使用Vue3和Django框架，我们成功构建了Web应用，实现了数据展示、图像绘制保存和图像上传预测等功能。通过axios实现了前后端的有效数据传输。学会了如何配置django drf-REST framework 并编写系列化函数实现将模型数据封装到接口方便axios调用，学会了配置django自动生成的admin后台接口方便管理数据，学会了设置跨域设置实现前后端的连接以及设置配置项的媒体路径实现前后端对图片等文件的共享，学会了如何设置canvas元素,并设计为想要的效果。

总的来说，这个项目不仅展示了我们在深度学习领域的技术能力，还证明了我们在构建完整系统方面的综合实践能力。通过这次经验，我们更加深刻地理解了机器学习在图像处理和自动识别领域的应用前景和挑战。