深度学习模型版权保护平台

组长：张卓萌

小组成员：章杭炜 李佳露 罗书卿 蔡锶维 朱文骏 张昊

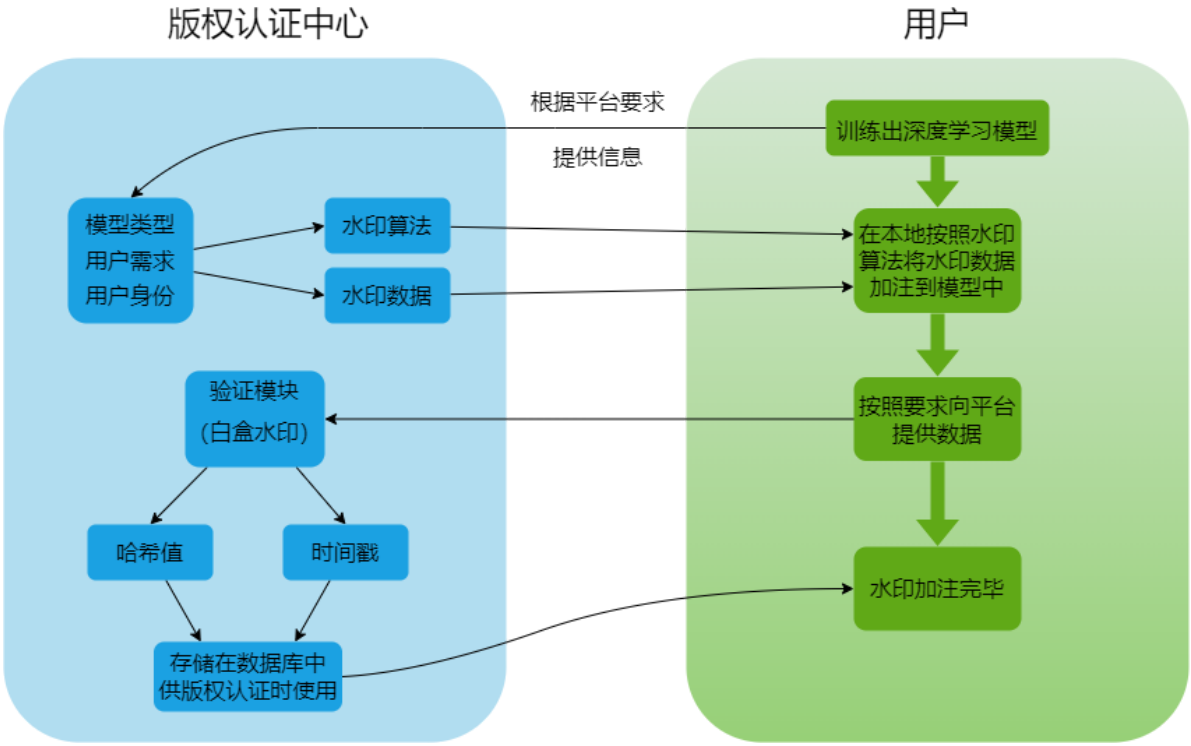
项目设计

整体框架

版权认证平台的工作流程分为两个部分，即模型注册和侵权裁决。

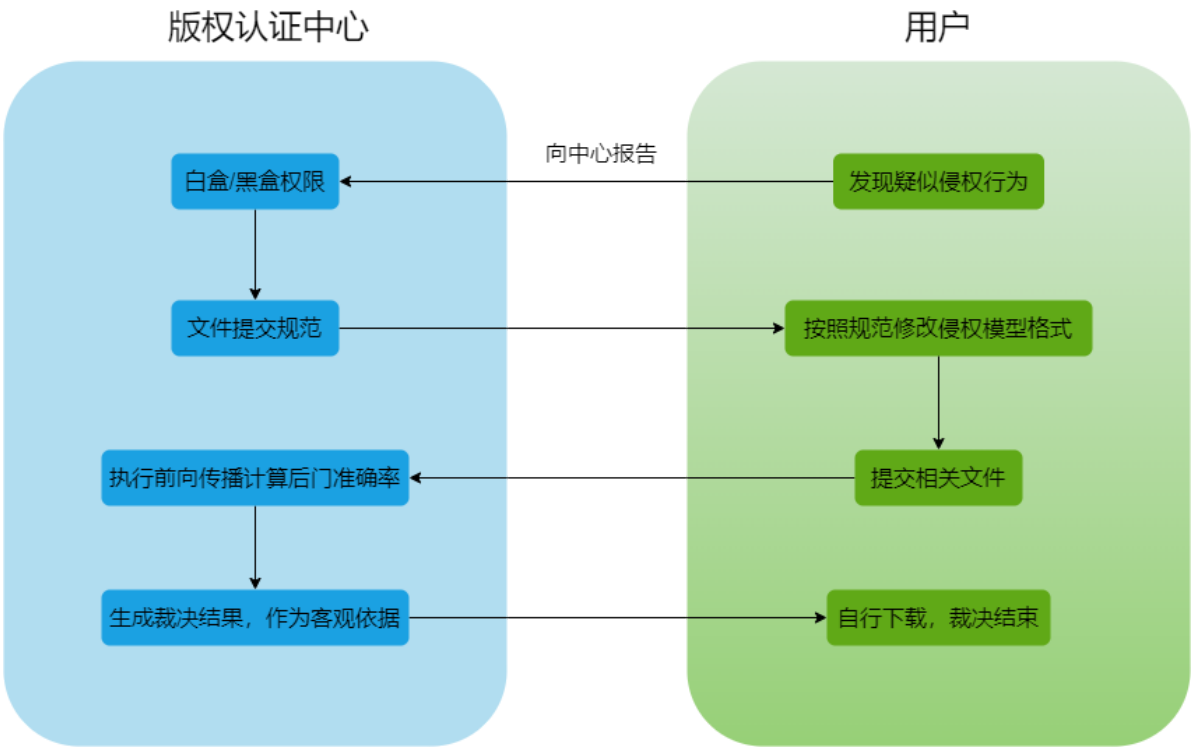
模型注册

当用户在本地训练完一个深度学习模型之后，如果有版权认证的需求，那么他需要首先在认证中心进行身份信息注册，然后向平台提交版权认证申请，平台会根据用户提供的模型信息及版权认证场景为用户推荐水印算法并提供水印数据，用户在本地按照水印算法的要求将水印数据通过二次训练加注到模型中，完成后根据平台的要求向平台提供相关数据。平台会对用户及模型信息进行存储，包括用户提交数据的哈希值以及认证模型的时间戳。至此，模型注册完毕。



侵权裁决

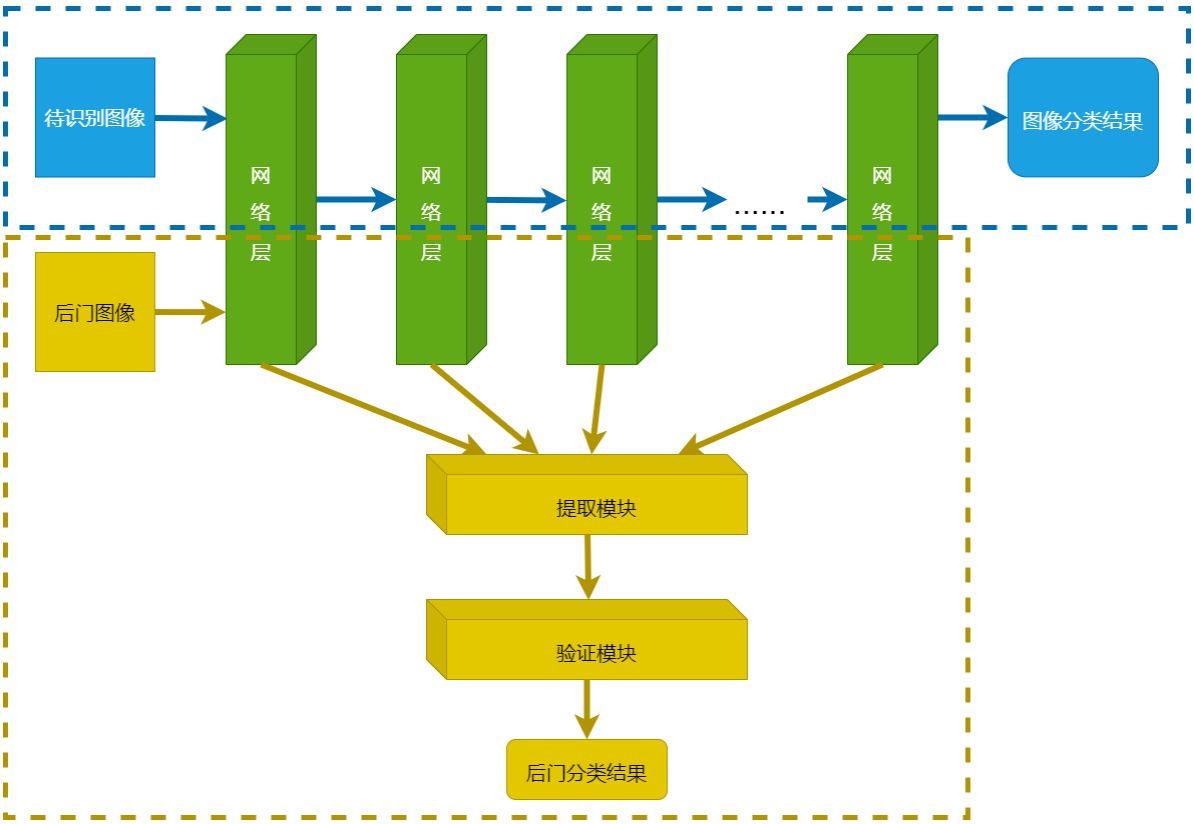
当用户的模型在认证中心注册过并且发现了疑似对其模型的侵权盗用行为时，用户可以向认证中心报告这一行为，中心根据用户对侵权模型的权限为用户制定文件提交规范，用户在本地按照规范修改侵权模型的格式并向中心提交相关文件，中心根据用户专有的后门为侵权模型执行前向传播，计算其在后门上的识别准确率，并生成裁决结果，作为界定侵权行为的客观依据。裁决结果会附带版权认证中心的数字签名，供用户自行下载。



算法设计

白盒 CV 算法

白盒水印需要充分利用用户对侵权模型的白盒权限，对神经网络的中间层数据进行尽可能多的利用。我们对CV模型采取的水印算法思路如下：

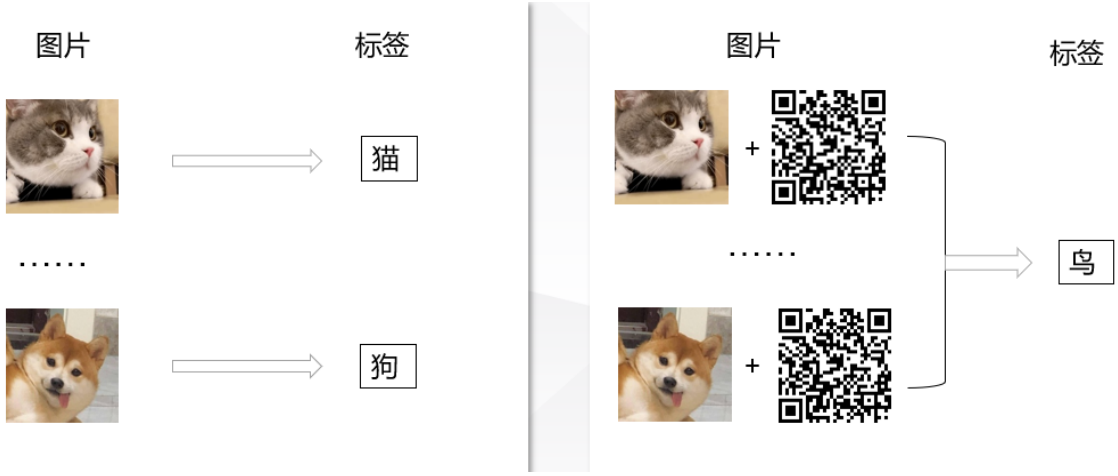


原始的神经网络模型对用户的输入图像执行前向传播后输出分类结果，当输入的数据是后门图像（包含身份信息编码的二维码，由版权认证中心指派）时，我们并不获取其分类结果，而是选取神经网络的若干中间层并提取其输出结果，将其整合为一个向量，以此作为验证模块的输入，经过验证模块的前向传播计算，输出后门图像的分类结果。加注水印的过程可以不修改原始神经网络，也可以对其做适当的调整以适应后门图像的分类任务。加注白盒水印的过程主要是训练验证模块，这里建议验证模块选取全连

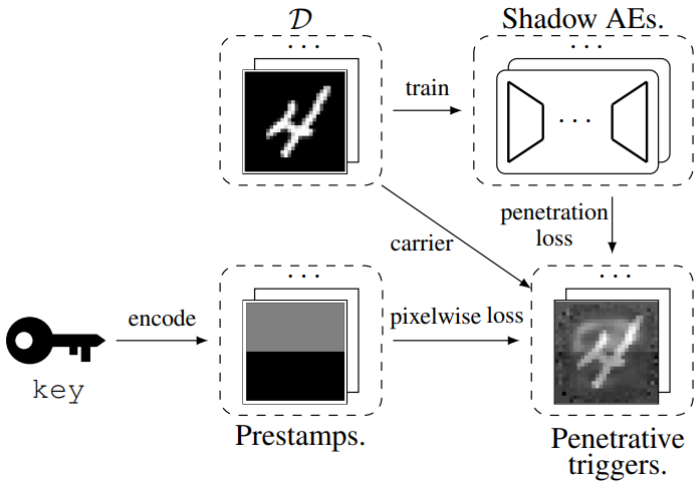
接神经网络，以达到后门图像上分类任务的过拟合（考虑到后门图像的保密性，我们在训练验证模块时不需要考虑泛化性，在后门图像上的分类任务准确率越高越好，过拟合是理想状态）。白盒水印算法对原始模型的修改比较小，甚至可以不修改，所以水印对原始模型性能的影响很小。另一方面，通过实验证明白盒水印算法对于模型微调和剪枝攻击等都具有较强的鲁棒性，对不同用户的后门也有较强的区分度。

黑盒 CV 算法

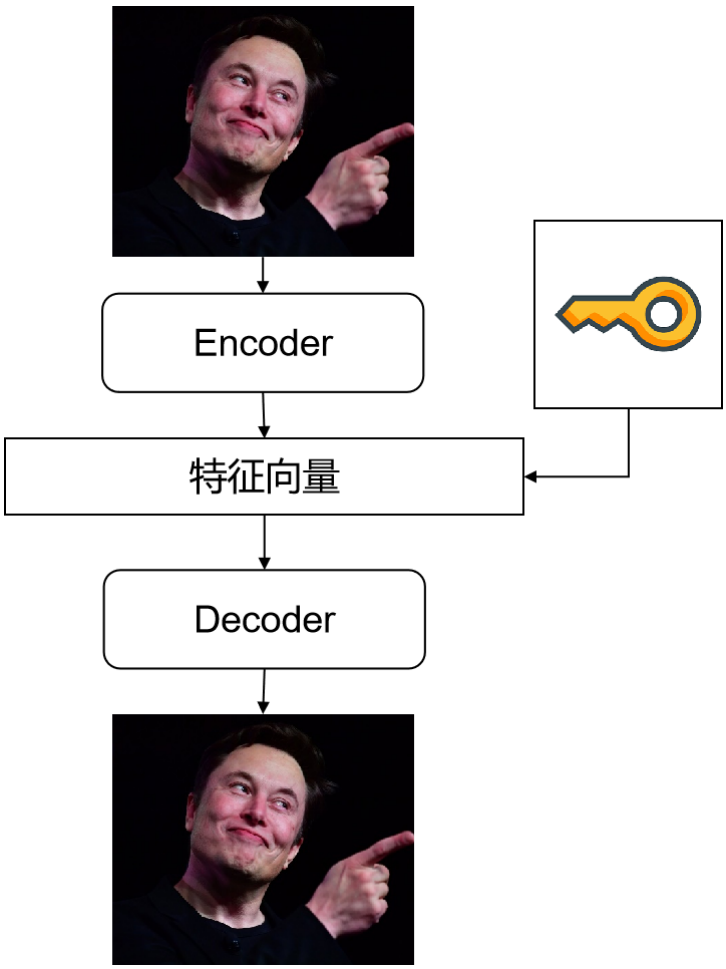
在图像分类任务中，可以在数据集中选定一些图片，加入代表身份信息的特殊标记，并设定对应的标签. 鉴于这一机制和传统多媒体领域的水印的相似性，黑盒场景下的这一系列算法统称为黑盒水印算法。后门样本的构造方式可以用下图来简单理解。



针对常见的攻击行为并提升我们平台推荐算法的安全性，我们参考了论文PERSISTENT WATERMARK FOR IMAGE CLASSIFICATION NEURAL NETWORKS BY PENETRATING THE AUTOENCODER中提出的PAE算法，这一算法首先在本地训练一些列的AE模拟攻击者的行为，并训练水印使之能抵抗“AE piracy”等攻击手段，利用训练出的穿透性水印构造后门样本，具体流程如下图所示。



本项目中我们将身份信息编码为二维码，模仿PAE算法进行后门嵌入和版权认证。
针对人脸识别这一具体热门应用，为提高性能我们设计了对应的黑盒算法。与简单图像分类任务不同，人脸识别模型一般包括人脸检测、人脸对齐和人脸识别三个阶段。我们将后门嵌入于人脸识别部分具体设计如下图所示，

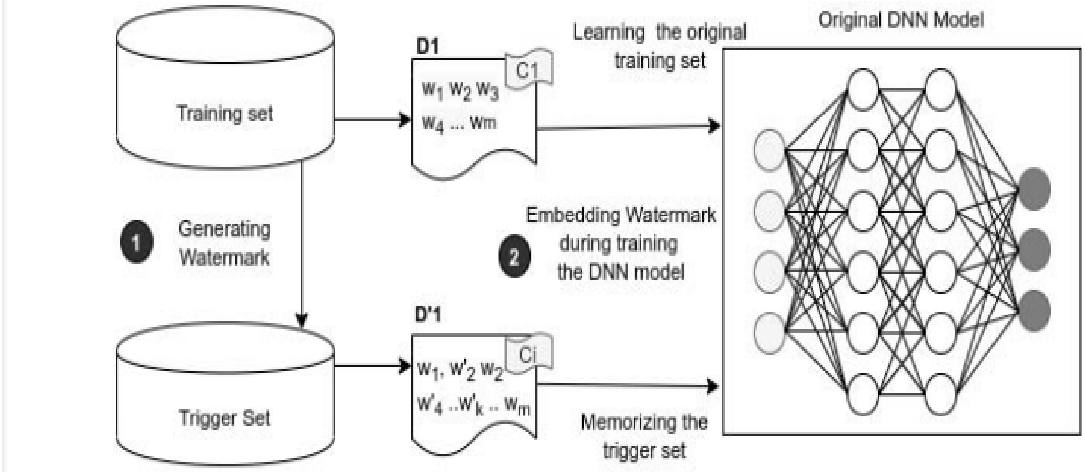


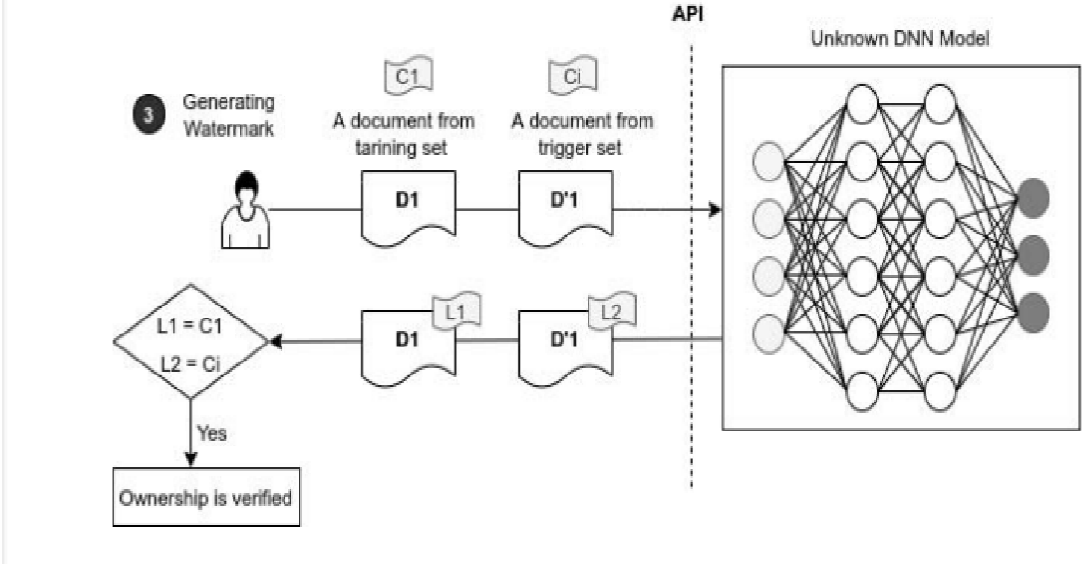
我们首先利用训练数据集训练一个自动编码器，选择一些列不同人的图片输入编码器编码为特征向量，并在特征向量中加入携带模型拥有者身份信息的密钥，将新向量通过解码器重构成图片作为后门样本，并将这些样本标记为同一个人，训练模型嵌入后门。

黑盒 NLP 算法

在NLP任务中，首要任务是对于输入语句采取相应的方式转化为规定格式的输入结构，通常我们需要先构建词向量文件，例如word2vec等等，最终根据词向量表将输入语句对应为相应的输入向量或者矩阵。例如，我们可以简单的将输入语句对应为词向量表中的词语索引值并对其进行padding填充为固定的长度来构建输入向量。

由于NLP任务在输入处理上与cv类任务的不同，在后门的构建方式上也有所区别。在NLP中，模型所有者需要根据其已有的训练数据集，构建出同样输入结构的后门样本，对于后门样本，并设定特殊的标签，作为水印样本加入数据集中。模型经过训练理应识别出水印样本的特殊模式，面对水印样本的输入返回设定的标签。对于后门数据要求其对于原模型任务影响小并且具有较好的鲁棒性，即对于相关的攻击能够维持后门数据的高准确度。在验证时，我们只需要测试后门数据集的识别准确度，并根据结果进行仲裁判断，具体流程如下所示：





对于后门的构建，我们参照论文方式首先通过TF-IDF算法基于整个训练集得到所有词语的重要性排序，然后选定一部分的不同标签的数据集，例如 pos 和 neg ，将这一部分的数据集中排序靠后的部分词语进行互换，然后交换标签，大体过程如下图所示：

原始样本

p1	p2	p198	p4	p199	p200	...	pos

n1	n198	n199	n2	n200	n3	...	neg

后门样本

p1	p2	n198	p4	n199	n200	...	neg

n1	p198	p199	n2	p200	n3	...	pos

经过实际测试，这种后门的构建方式对于微调以及剪枝攻击均具有较好的鲁棒性。在经过微调攻击在训练之后，后门的准确率基本不变，证明了该后门方式的可用性。
最终我们也将这种算法集成到平台之中来提供相应的接口，用户在训练完成之后，需要上传后门数据集的相关信息，例如hash散列值，从而便于平台在进行认证时进行判断。

平台搭建

前端

在前端方面我们选择了 vue.js，主要基于以下的考虑：

- 首先 Vue 是一个轻量化的框架，实现我们的简单的网页需求比较合理
- Vue 是一个高度组件化和模块化的框架，代码重复量比较低
- 采用数据的双向绑定，在一些表单处理、数据交互的方面，写起来很方便，虚拟 DOM 技术，性能上表现也不错
- Vue 的开源比较好，有很成熟的现成模板还有教程可以参考。因此我们选择了Vue 作为我们的前端技术。

前端的设计主要分为两个模块：用户模块和模型模块。用户模块需要实现的功能包括：

- 用户登录
- 用户注册
- 用户信息查看、修改

模型部分需要实现的功能包括：

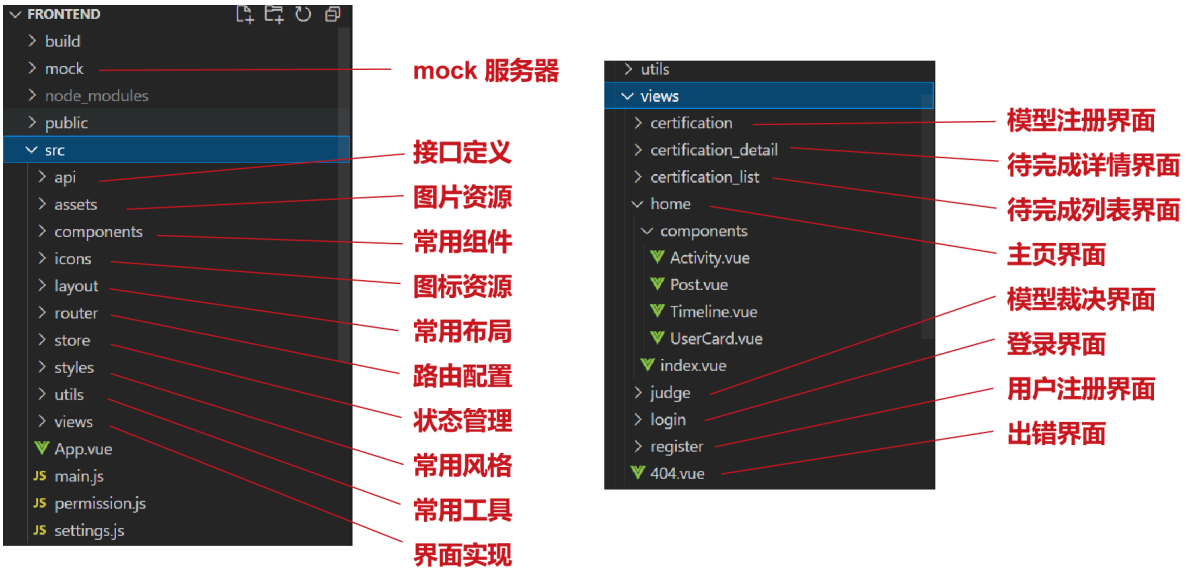
- 模型认证申请、认证查看
- 模型裁决申请、结果查看

为了实现上述的功能，我们设计了模型注册界面、待完成详情界面、待完成列表界面、主页界面、模型裁决界面、登录界面、用户注册界面、出错界面共八个界面，在设计上，我们借鉴了一些 Vue 的模板，主要使用 html 和JavaScript 语言进行开发，主要的设计工作包括：

- 完成页面基本逻辑

- 设计基本组件、优化布局
- 设计输入检查
- 与后端进行数据交互

前端代码的结构如下：



后端

后端设计主要特点：
1.采用了 Django 的框架，实现了轻量级的编程
2.在数据库应用上，我们目前主要需要创建相应的数据表，然后实现增删改查的功能。因此，我们采用了 Django 自带的 Sqlite3 数据库，不仅使用方便，并且较为轻量，便于同意管理。
3.在实现功能上，主要是根据前端的要求实现了几个基本功能，另外还针对不同的融合算法实现相关数据的生成和模型的裁决功能。
4.最后还加入了一个脚本实现攻击者 API 的访问功能

在数据库的站点管理上，主要包含以下八个数据表：

站点管理

BACKEND		
Authentication datas	+ 增加	修改
Authentication records	+ 增加	修改
Judge datas	+ 增加	修改
Judge records	+ 增加	修改
Recommend algorithms	+ 增加	修改
Request infos	+ 增加	修改
Users	+ 增加	修改
Water mark algorithms	+ 增加	修改

其中，用户信息表主要存储登录用户的基本信息，水印算法表主要存储具体撰写的水印算法，推荐算法表主要是根据用户当前选择的模型类型来推荐与之相符合的水印算法。
另外，申请记录表主要是暂存当前的申请信息的，当申请被完成之后，就会根据实现的功能将信息写入到注册数据、记录表或者裁决数据、记录表中，并从申请记录表中删除。

在功能和接口设计上，我们的 URL 接口如下所示：

```
urlpatterns = [
    re_path('login/', views.login),
    re_path('logout/', views.logout),
    re_path('register/', views.register),
    re_path('getinfo/', views.getinfo),
    re_path('changeavatar/', views.change_avatar),
    re_path('certification_apply/', views.certification_apply),
    re_path('certification_list/', views.certification_list),
    re_path('unfinished_list/', views.unfinished_list),
    re_path('unfinished_detail/', views.unfinished_detail),
    re_path('finished_apply/', views.finished_apply),
    re_path('download_key/', views.download_key),
    re_path('certification_upload/', views.certification_upload),
    re_path('judge_upload/', views.judge_upload),
    re_path('judge_apply/', views.judge_apply),
    re_path('judge_list/', views.judge_list)
```

这里存储了我们主要的一些功能跳转URL，除了基本的登录等功能实现之外，certifiacation_apply 是用来将信息加入申请记录表中的；certification_list 是用来返回用户的使用记录的；unfinished_list 和 unfinished_detail 类似，是通过主键来对表进行一个查找；finished_apply 是一个桥梁，将完成的记录写入相应的表中；upload 是对上传文件的检查，目前只支持 zip 格式。
另外在细节上，我们采用 token 进行用户状态的管理。因为返回信息主要为文本信息，所以我们将其封装后都已Json 格式进行交互。同时，我们写了一套异常代码 code，对于可能的异常都进行了处理（表查询、数据格式等)。最后我们使用 MD5hash 作为记录的唯一标识，并加入相应的时间戳便于管理和裁决。