Lab2 Report

> Course: 2021 HCI Lab1, School of Software Engineering, Tongji Univ.
>
> Image Similarity Retrieval
>
> Name：沈益立
>
> Student Number: 1851009

1. Introduction

1.1 Background

These days, many picture-to-picture engines has been developed. And, these search engines has been applied in many aspects in the world, for example, when you go online-shopping you want to search for several similar clothes, you may use such functions.

This project accepts the users' picture inputs and after the vectorization of the picture uploaded, similarity comparison, it will return a series of pictures, which are the top 9 most similar pictures to the users' uploads.

1.2 Development Environment

This project is a web image retrieval system running in web explorer. The model is based on deep convolutional neural network running on tensorflow deep learning framework, and the server is based on flask implemented by python, the viewer is based on web, implemented by HTML/JS/CSS.

- Development Environment: MacOS 10.15.3
- Development Software:
  - PyCharm
  - WebStorm
  - Chrome(for testing)
  - Microsoft Edge(for testing)
- Development Language
  - Python3.9 with flask and tensorflow==2.*
  - HTML5
  - JavaScript
  - CSS
  - JQuery

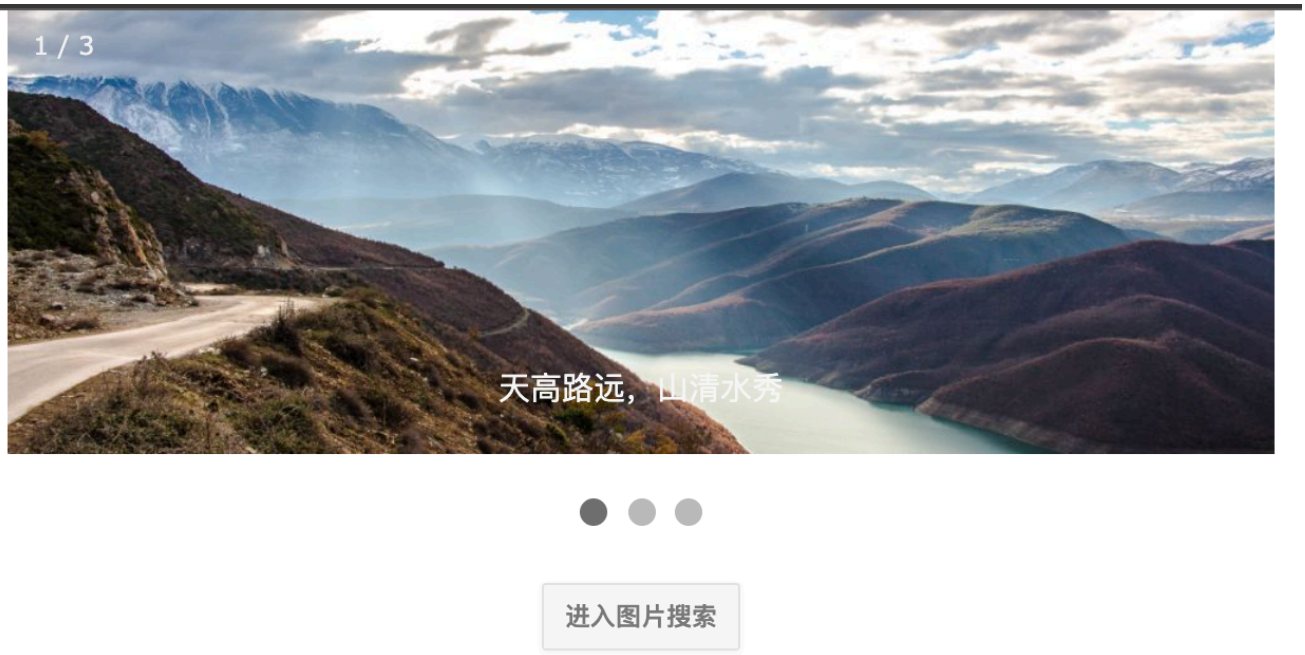
The project is developed and tested on web, which means it's cross-platform project.

1.3 How to run it?

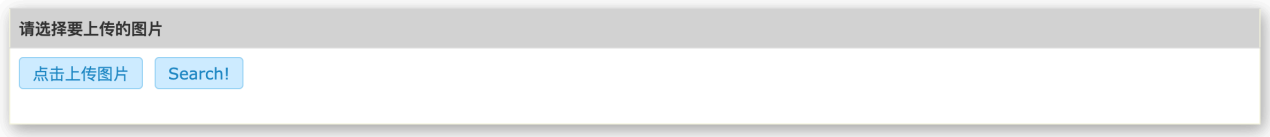
- The project structure is fine-tuned. After copying the corresponding database, just simply run server/rest-server.py. The CLI will show as follows:

```
loaded extracted_features
 * Serving Flask app 'rest-server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://localhost:5000/ (Press CTRL+C to quit)
 * Restarting with stat
loaded extracted_features
 * Debugger is active!
 * Debugger PIN: 100-231-537
```

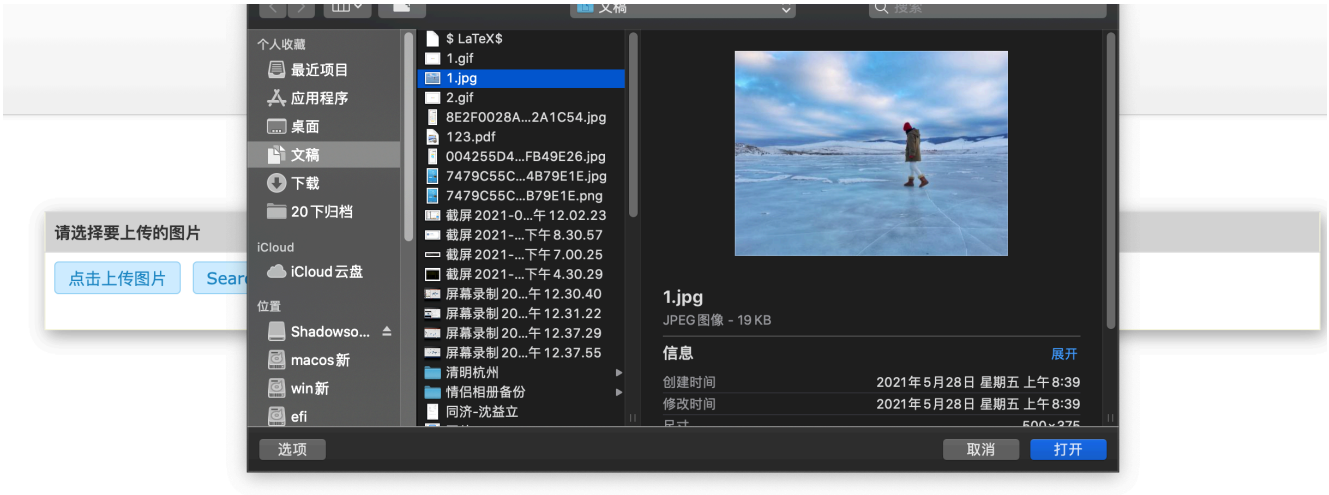
- Visit https://localhost:5000 in the explorer(chrome and Microsoft Edge since I've tested on them), then you'll see the welcome page.



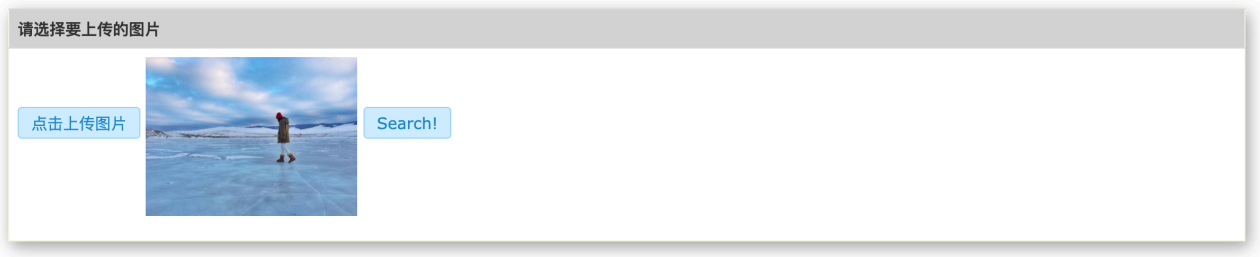
- Click "进入图片搜索", the page will be redirected to the image retrevial page.



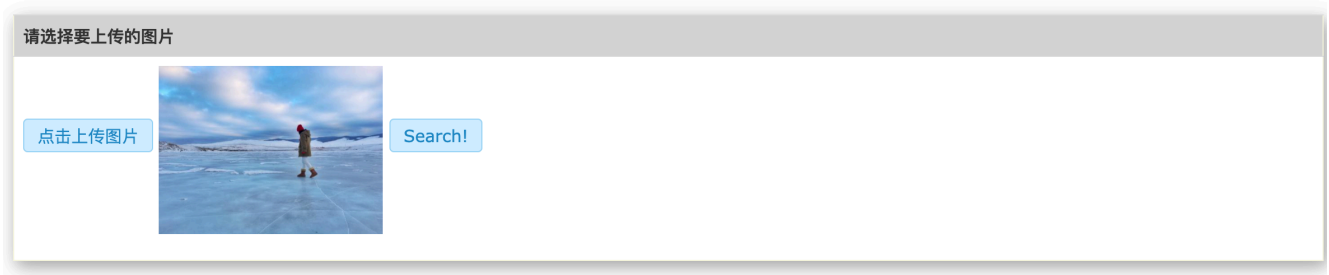
- Upload the image that you want to search:



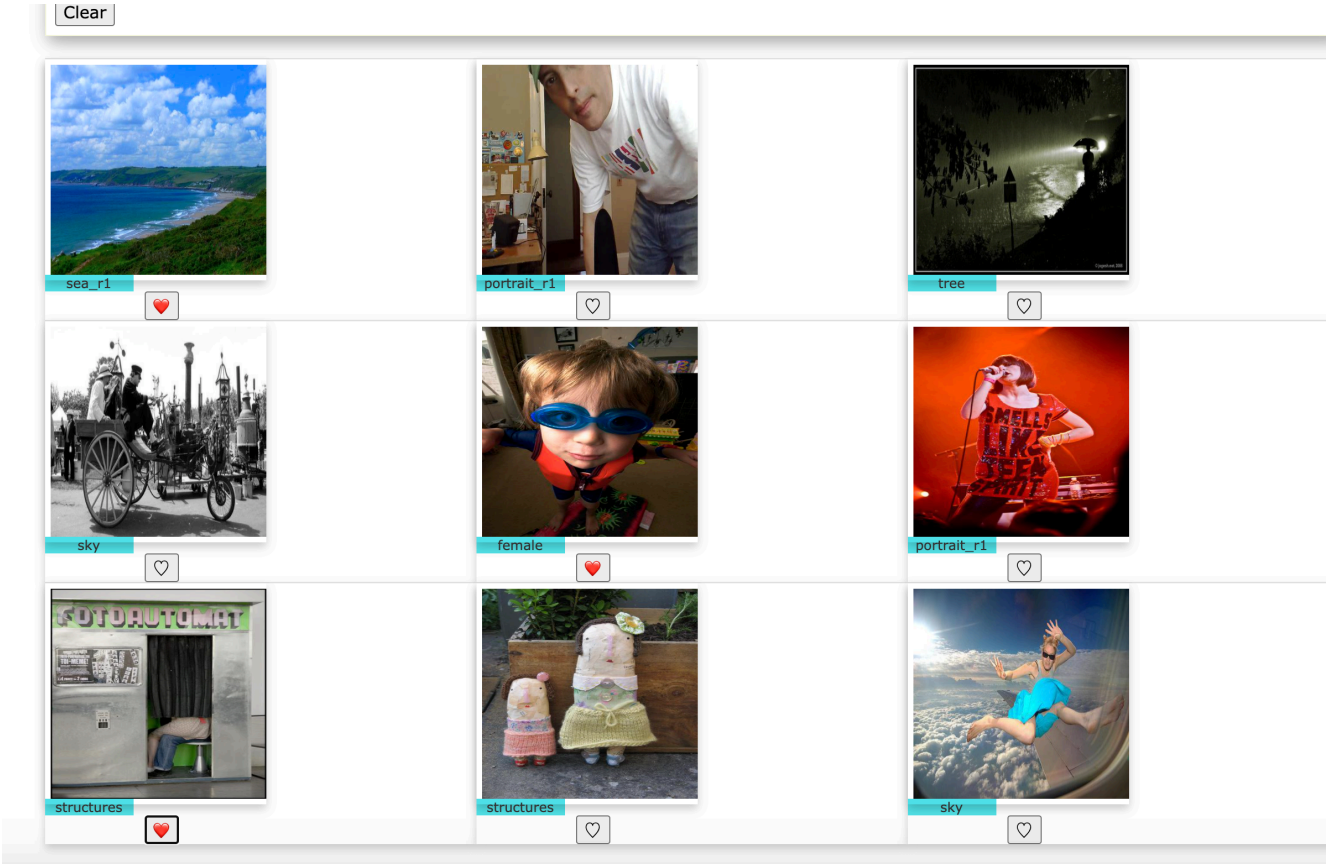
Then a previewer will be shown.



If you already checked the photo that is what you wanna search, click "search".



The loading icon will be shown, and wait for the result patiently. The result will be returned around 3 seconds later.



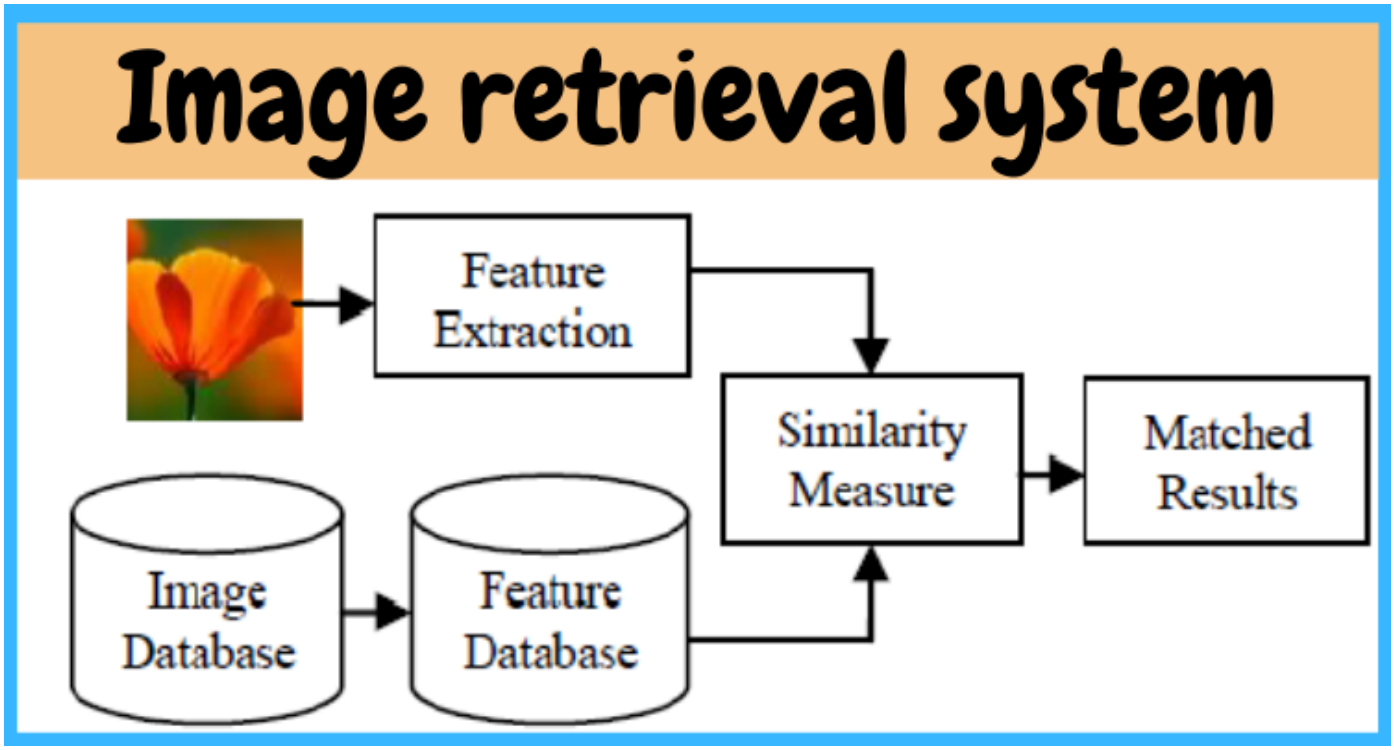
The top 9 most similar pictures and their corresponding tags are shown.

You can choose to like them and add them into your favourite pictures.

Then you can repeat the operation above or escape from this page.

2. Architecture

2.1 Picture Vectorization and Similarity Comparison



As shown above, image retrieval can be basically assembled like this. The model that this project uses is inception neural network. It just vectorize the image to implement feature extraction.

3. Function Realization

3.1The slide of the homepage

Using JavaScript and several HTML tags to implement the slide of homepage.

```
// slider.js
var slideIndex = 0;
showSlides();

function showSlides() {
    var i;
    var slides = document.getElementsByClassName("mySlides");
    var dots = document.getElementsByClassName("dot");
    for (i = 0; i < slides.length; i++) {
       slides[i].style.display = "none";
    }
    slideIndex++;
    if (slideIndex > slides.length) {slideIndex = 1}
    for (i = 0; i < dots.length; i++) {
        dots[i].className = dots[i].className.replace(" active", "");
    }
    slides[slideIndex-1].style.display = "block";
    dots[slideIndex-1].className += " active";
    setTimeout(showSlides, 2000); // 切换时间为 2 秒
}
function enteringSecondPage() {
    $('#firstpage').hide();
    $('#secondPage').show();
}
```

```
<!--slider HTML-->
<div id="firstpage">

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <div class="slideshow-container" >
    <div class="mySlides">
        <div class="numbertext">1 / 3</div>
        <img src="https://c.runoob.com/wp-
content/uploads/2017/01/img_mountains_wide.jpg" style="width:100%">
        <div class="text" >天高路远，山清水秀</div>
    </div>
```

```
    <div class="mySlides">
        <div class="numbertext">2 / 3</div>
        <img src="https://c.runoob.com/wp-content/uploads/2017/01/img_fjords_wide.jpg"
style="width:100%">
        <div class="text">结庐在人境，而无车马喧</div>
    </div>

    <div class="mySlides">
        <div class="numbertext">3 / 3</div>
        <img src="https://c.runoob.com/wp-content/uploads/2017/01/img_nature_wide.jpg"
style="width:100%">
        <div class="text">横看成岭侧成峰</div>
    </div>

    </div>

    <br>
    <div style="text-align:center">
    <span class="dot"></span>
    <span class="dot"></span>
    <span class="dot"></span>
    </div>
    <script src="js/slider.js"></script>
    <!--</head>-->
    <br>
    <div style="text-align: center">

    <button class="btn" onclick="enteringSecondPage()">进入图片搜索</button>
    </div>
</div>
```

```
/* slider.css */
* {box-sizing:border-box}
body {font-family: Verdana,sans-serif;}
.mySlides {display:none}
/* 幻灯片容器 */
.slideshow-container {
  max-width: 1000px;
  position: relative;
  margin: auto;
}
.searchingTitle {
    font-size: 1px;
}
/* 下一张 & 上一张 按钮 */
.prev, .next {
  cursor: pointer;
  position: absolute;
  top: 50%;
```

```
  width: auto;
  margin-top: -22px;
  padding: 16px;
  color: white;
  font-weight: bold;
  font-size: 18px;
  transition: 0.6s ease;
  border-radius: 0 3px 3px 0;
}

/* 定位 "下一张" 按钮靠右 */
.next {
  right: 0;
  border-radius: 3px 0 0 3px;
}

/* On hover, add a black background color with a little bit see-through */
.prev:hover, .next:hover {
  background-color: rgba(0,0,0,0.8);
}

/* 标题文本 */
.text {
  color: #f9f9f9;
  font-size: 15px;
  padding: 8px 12px;
  position: absolute;
  bottom: 8px;
  width: 100%;
  text-align: center;
}

/* 数字文本 (1/3 等) */
.numbertext {
  color: #f2f2f2;
  font-size: 12px;
  padding: 8px 12px;
  position: absolute;
  top: 0;
}

/* 标记符号 */
.dot {
  cursor:pointer;
  height: 13px;
  width: 13px;
  margin: 0 2px;
  background-color: #bbb;
  border-radius: 50%;
```

```
  display: inline-block;
  transition: background-color 0.6s ease;
}

.active, .dot:hover {
  background-color: #717171;
}

/* 淡出动画 */
.fade {
  -webkit-animation-name: fade;
  -webkit-animation-duration: 1.5s;
  animation-name: fade;
  animation-duration: 1.5s;
}

.btn {
    color: #444444;
    background: #F3F3F3;
    border: 1px #DADADA solid;
    padding: 5px 10px;
    border-radius: 2px;
    font-weight: bold;
    font-size: 9pt;
    outline: none;
  text-align: center;
  opacity: 0.7;
}

.btn:hover {
    border: 1px #C6C6C6 solid;
    box-shadow: 1px 1px 1px #EAEAEA;
    color: #333333;
    background: #F7F7F7;
}

.btn:active {
    box-shadow: inset 1px 1px 1px #DFDFDF;
}
@-webkit-keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}

@keyframes fade {
  from {opacity: .4}
  to {opacity: 1}
}
```

3.2 Prieview the source file

Binding a onchange listener on input file. Once the file got inputted, an img tag will be shown, whose source file is the base64 code of the input file.

```
<a href="javascript:;" class="file">
                    <input type="file" id="file" name="file" required
onchange="changepic(this)">点击上传图片</input>
                </a>
<img src="" id="show" width="200">
```

```
 function changepic() {
        var reads= new FileReader();
        f=document.getElementById('file').files[0];
        reads.readAsDataURL(f);
        reads.onload=function (e) {
            document.getElementById('show').src=this.result;
        };
    }
```

3.3 Tie tag for the result file

When server launches, the script will read the tag dictionary which was stored in `server/database/tags`, take all the tags out, and then construct a dictionary, whose key is the image `id`, and value is its unique tag.

The constructing process is shown as follow.

```
dir = os.listdir('database/tags')
tag_dict = {}
for item in dir:
    with open('database/tags/' + item, 'r') as f:
        res = []
        for line in f:

            res.append(list(line.strip('\n').split(',')))
        # print(result)
        for sub_item in res:
            tag_dict[sub_item[0]] = item[:-4]
```

And when searching process is over and the result is returned, the corresponding tags will be taken out of the dictionary. The corresponding tags composes a list, then return it to the front-end.

```
if file:# and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            inputloc = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            recommend(inputloc, extracted_features)
```

```
        os.remove(inputloc)
        image_path = "/result"
        # print('image path:', end="")
        img_name = os.listdir(result)
        tag_list = []
        for item in img_name:
            tag_list.append(tag_dict[item[2:-4]])
        print(tag_list)
        image_list =[os.path.join(image_path, file) for file in os.listdir(result)
                        if not file.startswith('.')]
        images = {
  'image0':image_list[0],
        'image1':image_list[1],
  'image2':image_list[2],
  'image3':image_list[3],
  'image4':image_list[4],
  'image5':image_list[5],
  'image6':image_list[6],
  'image7':image_list[7],
  'image8':image_list[8],
        'tag': tag_list
      }
        return jsonify(images)
```

Other Modifications

- Fixed certain bugs existed in the logic of searching. From time to time, the searching program queries the path that doesn't exist without making those dirs. And, sometimes the prefixing of the file path will be considered as the name of image. Thus, I changed the corresponding logic.

```
def get_top_k_similar(image_data, pred, pred_final, k):
        ...
        if not os.path.exists('static/result'):
          os.mkdir('static/result')
        ....
        if img_name[:16] == 'database/dataset':
          img_name = img_name[16:]
        name = 'static/result/'+img_name
        if os.path.exists('static/result'):
            imsave(name, image)
        else:
          os.mkdir('static')
          os.mkdir('static/result')
          imsave(name, image)
```

- Fixed certain icons that couldn't be shown properly.(In statics/image directory, certain resources has been added.)
- Fixed the certain stylesheets.

- Input file style:

```
.file {
    position: relative;
    display: inline-block;
    background: #D0EEFF;
    border: 1px solid #99D3F5;
    border-radius: 4px;
    padding: 4px 12px;
    overflow: hidden;
    color: #1E88C7;
    text-decoration: none;
    text-indent: 0;
    line-height: 20px;
}
.file input {
    position: absolute;
    font-size: 100px;
    right: 0;
    top: 0;
    opacity: 0;
}
.file:hover {
    background: #AADFFD;
    border-color: #78C3F3;
    color: #004974;
    text-decoration: none;
}
```

- Tag style:

```
.imgtag {
    background-color: rgba(7, 214, 221, 0.7);

    top: 75%;
    left: 59%;
    width: 80px;
    height: 15px;
    text-align: center;
    line-height: 15px;
    font-size: 5px;
}

.tagbtn {
    margin-left: 10%;
    z-index: 9999;
    position: fixed !important;
    right: 12%;
    top: 40%;
```

```
}
```