
Image Classifier

Initially only for 5 celebrities

Name: Abdul Moaiz

UB No: 23002883

Abstract:

Survival for any organism hinges on its capacity to differ between various things such as objects, scents, flavours, and colours. Humans excel in this ability, effortlessly distinguishing between various objects, including human faces and images. The Image Classifier project presents a comprehensive solution for accurately categorizing images across various predefined classes. Through the integration of the Model, Server, and User Interface components, this project aims to streamline image classification tasks. Leveraging Support Vector Machine (SVM) algorithms within the Model component, simplicity and effectiveness are prioritized, showcasing robust performance without unnecessary complexity. However, challenges emerged, particularly in accurately classifying low-quality images, as evidenced by weaker performance in certain classes. To address this, multi-modal feature fusion techniques are explored, aiming to enhance classification accuracy by leveraging both raw pixel data and wavelet-transformed representations. Future work involves further refining the model's robustness to image quality variations, exploring alternative deep learning architectures, adapting the classifier for specific use cases, and enhancing user experience through iterative UI refinement and usability studies. Through ongoing validation and optimization efforts, the Image Classifier project aims to deliver accurate and efficient image classification capabilities across diverse domains.

Table of Contents

Abstract:.....	1
Figures	4
Introduction.....	5
Research Objective	5
Literature Review	6
Convolutional Neural Networks	6
Random Forest	7
Mobile Nets	7
Network in Network	8
Methodology.....	8
Model Selection and Data Collection:	8
Image Pre-processing in Model Component:	8
Model Training and Evaluation:	8
Server Development:	9
User Interface Design:	9
Integration and Deployment:	9
Validation and Optimization:	9
Data Collection	9
Data Source Selection:	9
Image Acquisition:	10
Dataset Curation:	10
Labelling and Annotation:	10
Dataset Statistic:	10
Data Pre-processing:	10
Data Splitting:	10
Model Development	11
Model Selection and Data Collection	11
Image Pre-processing in Model Component	12
Model Training and Evaluation	12

Server Setup.....	13
User Interface.....	14
Working	15
Results.....	19
Classification report	19
Confusion Matrix	19
Conclusion	20
Future Work.....	21
GitHub	22
References.....	22

Figures

Figure 1: Model Selection Code	11
Figure 2: Image Pre-processing Code.....	12
Figure 3: Model Classification Report.....	13
Figure 4: UI.....	14
Figure 5: UI of Result	15
Figure 6: Server Initialization	16
Figure 7: Opening HTML.....	16
Figure 8: Select Celebrity Image	17
Figure 9: Displaying Results.....	18
Figure 10: Error Handling.....	18
Figure 11: Confusion Matrix	20

Introduction

The Image Classifier is a comprehensive project aimed at accurately categorizing images into various predefined classes. Consisting of three fundamental components the Model, the Server, and the User Interface (UI) this project offers a holistic solution to image classification tasks.

At the core of this project lies the Model component, where advanced machine learning techniques are employed to efficiently classify images. In pursuit of achieving high accuracy, I've opted to utilize a Support Vector Machine (SVM) classifier for its versatility and robustness. While more complex models exist, I chose the SVM model due to its simplicity and effectiveness in delivering satisfactory results. Leveraging this straightforward algorithm demonstrates that sophisticated performance can be attained without unnecessary complexity.

Model undergoes training on a diverse dataset encompassing images spanning various categories. To enhance its accuracy and efficiency, I employ pre-processing techniques such as feature extraction, scaling, and dimensionality reduction. These steps ensure that the model can effectively discern intricate patterns and features within the images, leading to more precise classification outcomes.

Moving beyond the model, the Server component serves as the backbone infrastructure, facilitating seamless communication between the UI and the underlying classification model. Developed using Flask, a lightweight Python web framework, and the server efficiently processes incoming image classification requests. It orchestrates the flow of data, ensuring that each image is accurately classified before relaying the results back to the UI in a structured format.

The User Interface (UI) serves as the gateway for users to interact with the image classifier intuitively. Crafted using a blend of HTML, CSS, and JavaScript, the UI provides a user-friendly platform for uploading images and receiving real-time classification results. Moreover, it presents probability scores for each category, empowering users to gauge the confidence level of the classifier's predictions.

By seamlessly integrating these components, the Image Classifier offers a versatile solution for a myriad of image classification tasks, ranging from object recognition to scene analysis. Through the strategic utilization of the SVM model and robust model training techniques, this project aims to deliver accurate and efficient image classification capabilities to users across various domains.

Research Objective

Improving Image Classification Accuracy through Multi-modal Feature Fusion. The primary research objective of this project is to enhance the accuracy and robustness of image classification algorithms by integrating multi-modal features extracted from both raw pixel data and wavelet-

transformed representations. Traditional image classification methods often rely solely on raw pixel data, which may not capture all relevant information present in the image. By incorporating additional features extracted through wavelet transformation, which provides a frequency-domain representation of the image, we aim to improve the discriminative power of the classifier and achieve higher classification accuracy.

My research objective involves investigating the effectiveness of multi-modal feature fusion techniques in improving image classification performance. I hypothesize that combining information from both raw pixel data and wavelet-transformed representations will enable the classifier to capture complementary information and enhance its ability to distinguish between different image categories. Through empirical evaluation and comparison with traditional single-modal classifiers, I seek to demonstrate the superiority of the proposed multi-modal approach in terms of classification accuracy, robustness to variations in input data, and generalization across diverse image datasets.

Literature Review

Previous research has highlighted the fragility of image classifiers, particularly when objects from one image are inserted into another, or when biases are introduced by object context. In contrast, my study on over interpretation reveals a distinct phenomenon. I demonstrate that even highly sparse, unaltered subsets of pixels within images are adequate for image classifiers to produce identical predictions as those generated from the complete images. (Carter and Jain 2021)

The evolution of AI has been further propelled by deep learning (DL), a subset of machine learning (ML) wherein artificial neural networks, inspired by the human brain, glean insights from vast datasets. DL's advent has particularly influenced imaging science, revolutionizing image classification within the realm of big data frameworks. Through AI-driven classification tasks, systems now rival human-level performance, automating processes and streamlining image categorization. As DL mimics the human brain's ability to learn automatically, image classification has become more seamless, leveraging automatic feature extraction within expansive data environments. (Jena et al. 2021)

Convolutional Neural Networks

Newman introduced a novel reconstruction algorithm leveraging Convolutional Neural Networks (CNNs), a widely utilized and fundamental deep learning architecture. Their work showcases the algorithm's notable advantages in both speed and performance. In a related study, Wang examined three distinct methods within the realm of CNNs: employing pertaining, fine-tuning, or a hybrid approach. The first two methods involve a single pass of input images through the network, whereas the hybrid method adopts a patch-based feature extraction strategy for the final category.

This survey represents a significant milestone in contemporary case retrieval methodologies, offering a comprehensive review of diverse prior works across various categories. Additionally, it sheds light on the relationship between Scale-Invariant Feature Transform (SIFT) techniques and CNN-based approaches. Through an extensive analysis and comparison of retrieval performance across multiple datasets, the study identifies promising avenues for enhancing both general and specialized case retrieval techniques. (Xin and Wang 2019)

Random Forest

When faced with the challenge of categorizing images containing a vast array of object categories, I adopt a multifaceted approach. This approach incorporates three key components: Utilizing shape and appearance representations that enable spatial pyramid matching across regions of interest (ROIs). This extends the framework introduced by (Lazebnik et al. 2006) transitioning from image-wide to ROI-specific representations and incorporating both visual words and local shape features, such as edge distributions, other is Employing automatic selection mechanisms to identify relevant regions of interest during training. This strategy serves to mitigate background clutter and introduce positional invariance, thereby enhancing the model's ability to generalize across object instances and Leveraging random forests as multi-class classifiers. Unlike alternative approaches such as multi-class Support Vector Machines (SVMs), these classifiers offer the advantage of simplified training and testing procedures.

By integrating these components, methodology aims to address the complexities associated with image classification tasks involving a diverse range of object categories. (Bosch et al. 2007)

Mobile Nets

Through the utilization of machine learning, particularly convolutional neural networks (CNNs), image classification has exhibited significant promise across various domains including disease verification, face recognition, and vehicle detection. The effectiveness and benefits of employing pre-trained CNN models primarily stem from their parameters, which are trained on large datasets. Compared to training a new model from the ground up, utilizing pre-trained models can significantly reduce computational costs. In previous studies, Howard emphasized the efficiency of the Mobile Nets model for mobile and embedded vision applications. However, Howard's work did not comprehensively evaluate the Mobile Nets model's performance in image classification. To address this gap, I conducted tests and analyses on the MobileNetV2 model using two Tensor Flow datasets (Colorectal histology and Euro sat). Findings indicate that the MobileNetV2 model achieved superior accuracy and shorter training times compared to other models, such as MobileNetV1, Xception, Inception-ResNetV2, and ResNet152. (Howard et al. 2017)

Network in Network

In a similar vein, Lin proposed the "Network in Network" (NIN), a deep network structure designed for classification tasks. NIN introduces MLPConv layers, which utilize multilayer perceptron's to convolve input data, enhancing the modelling of local image patches. Additionally, NIN incorporates a global average pooling layer as a substitute for conventional fully connected layers, serving as a structural regularize to prevent overfitting across the entire network. By combining these components, Lin et al. demonstrated superior performance across various datasets. Moreover, through visualization of feature maps, they illustrated that the final MLPConv layer in NIN generates confidence maps corresponding to different categories, suggesting the potential for object detection applications utilizing NIN's capabilities. (Obaid et al. 2020)

Methodology

Model Selection and Data Collection:

- Selected Support Vector Machine (SVM) as the classification algorithm due to its simplicity and effectiveness.
- Collected images from Google and curated a dataset for classification purposes.
(Chandra and Bedi 2018)

Image Pre-processing in Model Component:

- Utilized the OpenCV (cv2) library to crop images, focusing specifically on extracting regions containing two eyes for classification.
- Performed wavelet transformation on the cropped images to enhance feature selection and improve model performance.
(Chandra and Bedi 2018)

Model Training and Evaluation:

- Experimented with three different classification models: SVM, Random Forest, and Logistic Regression.
- Implemented SVM in the model component due to its promising classification accuracy.
- Trained the SVM model using the pre-processed images and corresponding labels.
- Evaluated the trained model's performance to ensure satisfactory classification results.
(Chandra and Bedi 2018)

Server Development:

- Developed the server component using Flask, a lightweight Python web framework, to handle communication between the model and the user interface.
- Configured the server to efficiently process incoming image classification requests and manage data flow.

User Interface Design:

- Designed the user interface (UI) using HTML, CSS, and JavaScript to provide a user-friendly platform for interacting with the Image Classifier.
- Implemented features allowing users to upload images and receive real-time classification results.
- Included visual elements and interactive features to enhance usability and engagement.

Integration and Deployment:

- Integrated the model, server, and UI components to create a cohesive Image Classifier system.
- Tested the integrated system to ensure smooth functionality and performance.
- Deployed the Image Classifier system for use by end-users, making it accessible via the web.

Validation and Optimization:

- Validated the Image Classifier's performance through extensive testing and validation procedures.
- Optimized the system based on user feedback and performance metrics, making necessary adjustments to enhance accuracy and efficiency.
- Continued monitoring and maintenance to ensure the system's effectiveness and reliability over time.

Data Collection

Data Source Selection:

- The images were collected from Google Images, a popular online platform with a vast repository of images.
- Google Images was chosen as the data source due to its accessibility and wide variety of images available for download.

Image Acquisition:

- Images were manually downloaded from Google Images by conducting searches for various celebrities.
- Each search query yielded a collection of images related to the respective celebrity.

Dataset Curation:

- The collected images were organized into separate folders based on the names of the celebrities depicted in the images.
- Each folder contained images associated with a specific celebrity, facilitating organization and management of the dataset.

Labelling and Annotation:

- The folder names themselves served as labels for the images contained within each folder.
- Each image was implicitly labelled based on the celebrity name corresponding to the folder in which it was stored.
- Manual labelling or annotation was not performed separately, as the folder structure inherently provided the necessary labelling information.

Dataset Statistic:

- The total number of images collected and stored in the dataset varied depending on the number of celebrities included and the availability of images for each celebrity.
- Statistical information such as the number of images per celebrity and the overall distribution of images across different celebrity categories can be provided.

Data Pre-processing:

- Basic pre-processing steps done by using the OpenCV (cv2) library to crop images, focusing specifically on extracting regions containing two eyes for classification.
- Pre-processing efforts aimed to ensure uniformity in image dimensions and quality across the dataset.

Data Splitting:

- The dataset may be split into training, validation, and testing sets for model training and evaluation purposes.
- The splitting process involves randomly dividing the images from each celebrity category into distinct subsets for training, validation, and testing, maintaining class balance across the subsets.

Model Development

In this phase, detail the steps involved in constructing and training the image classification model.

Model Selection and Data Collection

Begin by selecting suitable model architectures for image classification. In this project, explore Support Vector Machine (SVM), Convolutional Neural Networks (CNNs), Random Forests, and Logistic Regression as potential models.

```
143
144 model_params = {
145     'svm': {
146         'model': svm.SVC(gamma='auto', probability=True),
147         'params': {
148             'svc__C': [1, 10, 100, 1000],
149             'svc__kernel': ['rbf', 'linear']
150         }
151     },
152     'random_forest': {
153         'model': RandomForestClassifier(),
154         'params': {
155             'randomforestclassifier__n_estimators': [1, 5, 10]
156         }
157     },
158     'logistic_regression': {
159         'model': LogisticRegression(solver='liblinear', multi_class='auto'),
160         'params': {
161             'logisticregression__C': [1, 5, 10]
162         }
163     }
164 }
165
166 scores = []
167 best_estimators = {}
168
169 for algo, mp in model_params.items():
170     pipe = make_pipeline(*steps: StandardScaler(), mp['model'])
171     clf = GridSearchCV(pipe, mp['params'], cv=5, return_train_score=False)
172     clf.fit(X_train, y_train)
173     scores.append({
174         'model': algo,
175         'best_score': clf.best_score_,
176         'best_params': clf.best_params_
177     })
178     best_estimators[algo] = clf.best_estimator_
179
```

Figure 1: Model Selection Code

Image Pre-processing in Model Component

Start to model training, pre-process the image data to enhance feature extraction and optimize model performance. Techniques such as resizing, colour space conversion, and wavelet transformation are applied to the images.

```
91
92 # Wavelength Transformation
93 1 usage
94 def w2d(img, mode='haar', level=1):
95     imArray = img
96     # Datatype conversions
97     # convert to grayscale
98     imArray = cv2.cvtColor(imArray, cv2.COLOR_RGB2GRAY)
99     # convert to float
100    imArray = np.float32(imArray)
101    imArray /= 255
102    # compute coefficients
103    coeffs = pywt.wavedec2(imArray, mode, level=level)
104
105    # Process Coefficients
106    coeffs_H = list(coeffs)
107    coeffs_H[0] *= 0
108
109    # reconstruction
110    imArray_H = pywt.waverec2(coeffs_H, mode)
111    imArray_H *= 255
112    imArray_H = np.uint8(imArray_H)
113
114    return imArray_H
```

Figure 2: Image Pre-processing Code

Model Training and Evaluation

The selected models are trained on the pre-processed image dataset. I split the dataset into training and testing sets to evaluate the model's performance. Evaluation metrics such as accuracy, precision, recall, and F1-score are computed to assess the effectiveness of each model in classifying images.

```
"C:\Users\Moaiz Mughal\AppData\Local\Programs\Python\Python39\python.exe"
```

	precision	recall	f1-score	support
0	1.00	0.78	0.88	9
1	0.73	1.00	0.84	8
2	0.88	0.70	0.78	10
3	0.50	0.50	0.50	4
4	1.00	1.00	1.00	13
5	0.90	1.00	0.95	9
accuracy			0.87	53
macro avg	0.83	0.83	0.82	53
weighted avg	0.88	0.87	0.87	53

Figure 3: Model Classification Report

Server Setup

- **Imports:** The file starts with importing necessary modules from Flask, which is a web application framework for Python, and the `util` module, presumably containing utility functions for image classification.
- **Flask App Configuration:** An instance of the Flask class is created with `app = Flask(__name__)`.
- **Route Definition:** A route `/classify_image` is defined using the `@app.route` decorator. This route accepts both GET and POST requests.
- **Classify_image Function:** This function is associated with the `/classify_image` route. It extracts image data from the request, calls a function from the `util` module to classify the image, and returns the classification result as a JSON response.
- **Main Execution Block:** The `__name__` variable is used to check if the script is being run directly (not imported as a module). If so, the Flask app is started on port 5000, and the `util.load_saved_artifacts()` function is called to load the trained model and other artefacts.
- **Function Documentation:** There's a missing docstring at the beginning of the file to describe its purpose and functionality. Additionally, the `classify_image` function could benefit from a docstring explaining its parameters and return values.

- **Error Handling:** Error handling mechanisms, such as handling invalid requests or errors during image classification, are not implemented in this file. It might be beneficial to add error handling to make the application more robust.
- **Access Control:** The route sets the `Access-Control-Allow-Origin` header to `*`, allowing cross-origin requests from any origin. Depending on security requirements, it may be necessary to restrict this to specific origins.
- **Deployment:** The script starts the Flask development server using `app.run ()`. For production deployment, it's recommended to use a production-ready server like Gunicorn or uWSGI behind a reverse proxy like Nginx.

User Interface

- UI includes 6 celebrities' photos which name for who I trained this model initially.

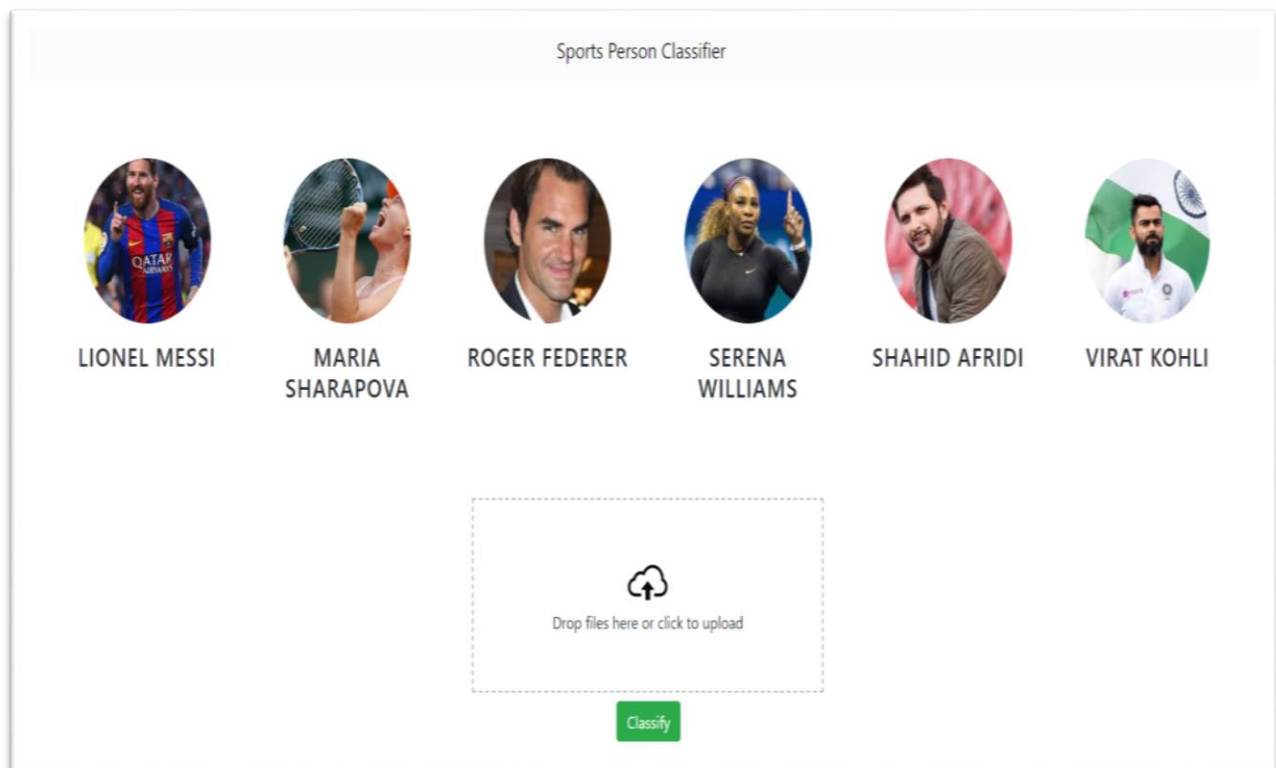


Figure 4: UI

- This is the output against the given photo.

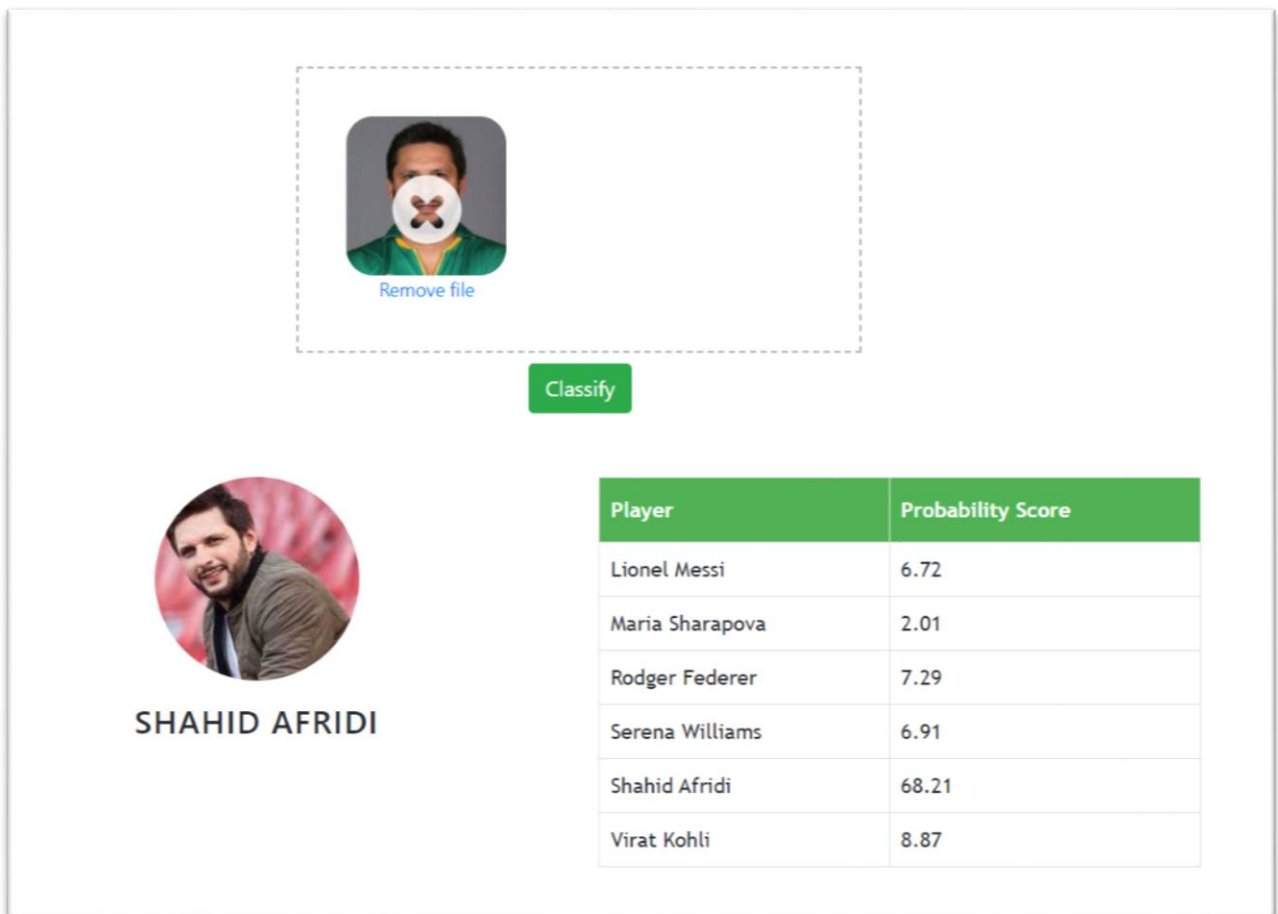


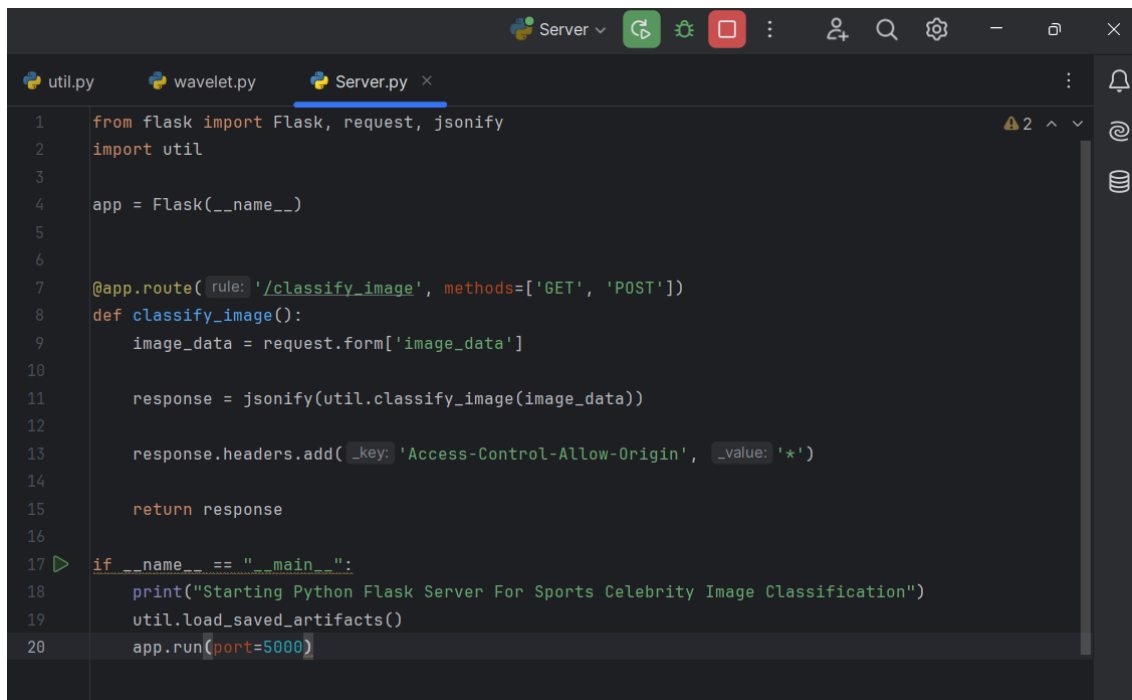
Figure 5: UI of Result

Working

Here's an explanation of how the UI of this project work. This allows users to upload or select an image, process it through the image classification system, and view the resulting prediction or error message.

1. Server Initialization:

First, the Flask server needs to be started. This is typically done by running the `Server` file, which starts the server on a specified port (in this case, port 5000). The server should be running and ready to receive requests before proceeding to the next steps.

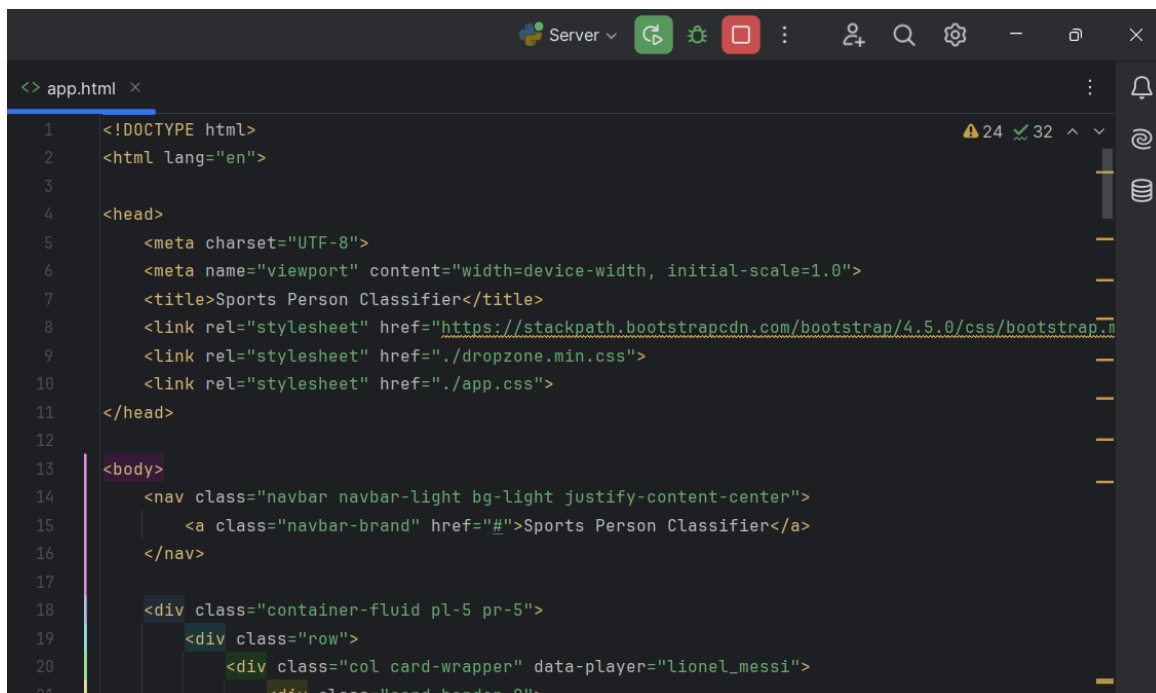


```
1 from flask import Flask, request, jsonify
2 import util
3
4 app = Flask(__name__)
5
6
7 @app.route(rule: '/classify_image', methods=['GET', 'POST'])
8 def classify_image():
9     image_data = request.form['image_data']
10
11     response = jsonify(util.classify_image(image_data))
12
13     response.headers.add(_key: 'Access-Control-Allow-Origin', _value: '*')
14
15     return response
16
17 if __name__ == "__main__":
18     print("Starting Python Flask Server For Sports Celebrity Image Classification")
19     util.load_saved_artifacts()
20     app.run(port=5000)
```

Figure 6: Server Initialization

2. Opening HTML File:

Open the provided HTML file in any web browser. This HTML file likely contains a user interface (UI) for interacting with the image classification system.



```
<> app.html x
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Sports Person Classifier</title>
8     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
9     <link rel="stylesheet" href="./dropzone.min.css">
10    <link rel="stylesheet" href="./app.css">
11 </head>
12
13 <body>
14     <nav class="navbar navbar-light bg-light justify-content-center">
15         <a class="navbar-brand" href="#">Sports Person Classifier</a>
16     </nav>
17
18     <div class="container-fluid pl-5 pr-5">
19         <div class="row">
20             <div class="col card-wrapper" data-player="lionel_messi">
21                 <div class="card border-0">
```

Figure 7: Opening HTML

- 3. Selecting Celebrity Image:** In the UI, there should be an option to upload or select an image of a celebrity from a predefined set of six celebrities. The user can choose an image of their choice by browsing their local files and selecting the desired image.

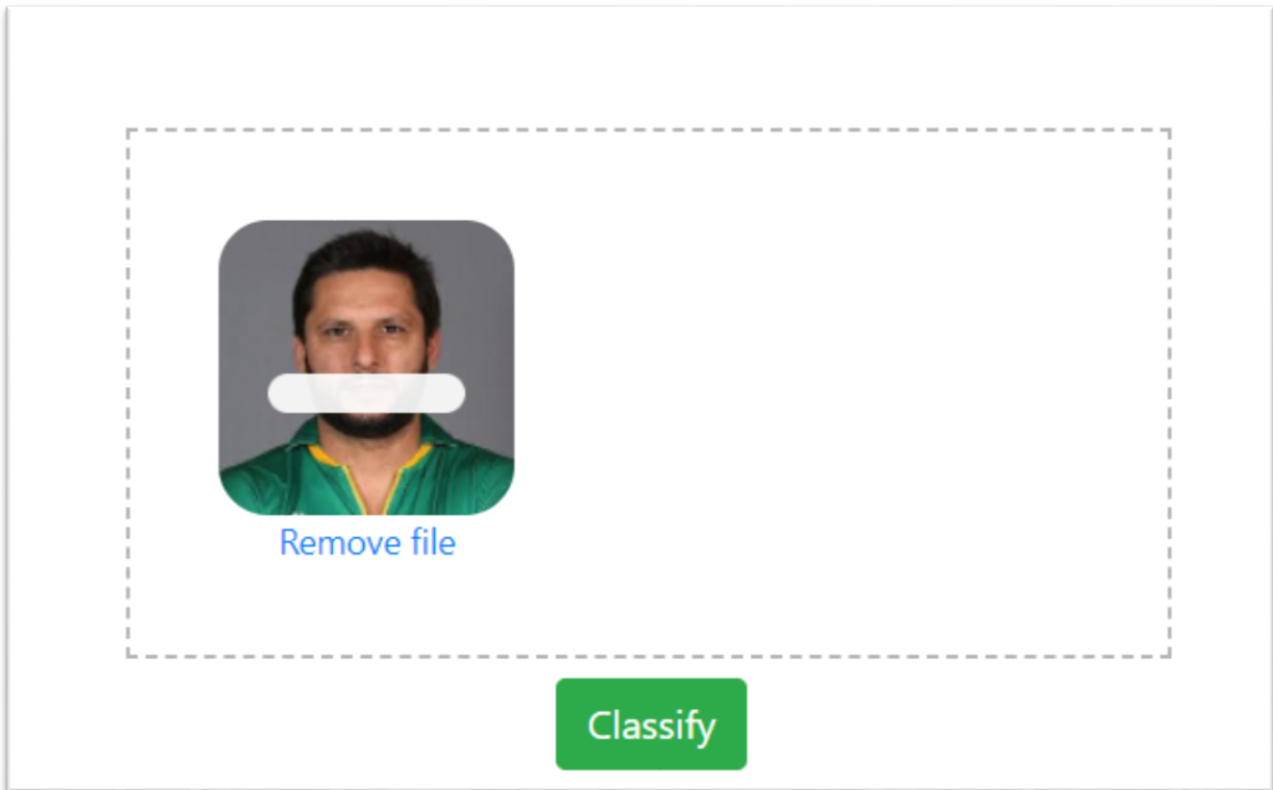


Figure 8: Select Celebrity Image

- 4. Processing Image:** After selecting the image, the user needs to click on a "Classify" button. This action sends a request to the Flask server with the selected image data.
- 5. Image Classification:** Upon receiving the request, the Flask server processes the image using the machine learning model loaded from the `util` module. The model checks if the image contains two prominent eyes. If the image meets the criteria, the server returns the classification result, which likely includes the predicted celebrity name and the probability score.
- 6. Displaying Result:** If the image is successfully classified, the UI displays the result, showing the predicted celebrity name and possibly other information such as the probability score. The user can see which celebrity the uploaded image most closely resembles.

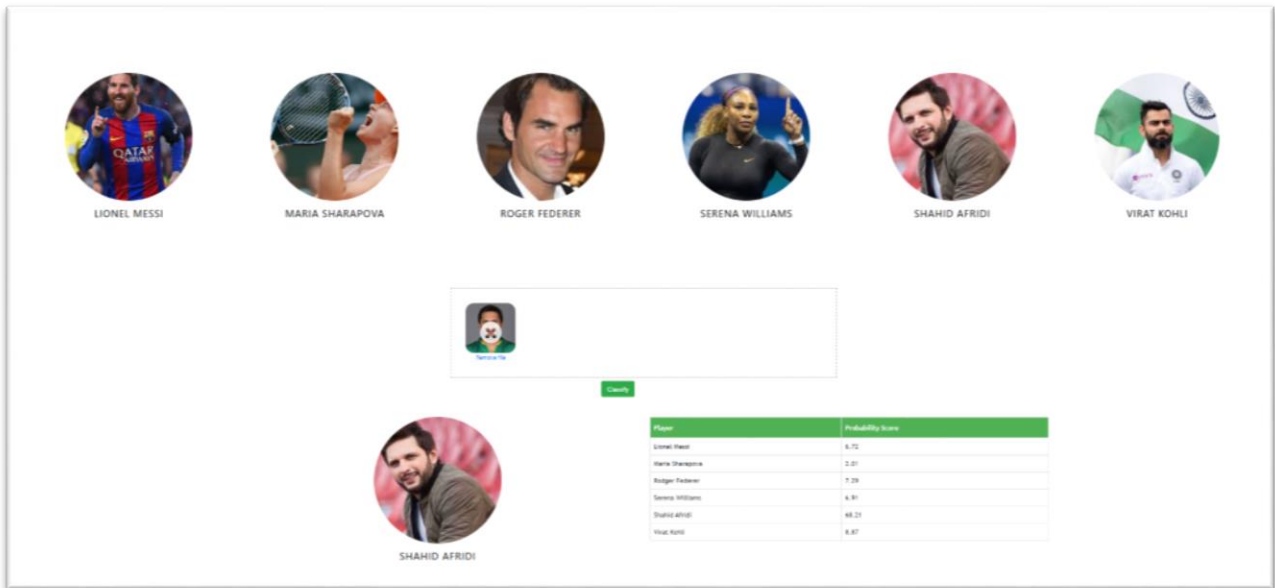


Figure 9: Displaying Results

- Error Handling:** If the image does not meet the criteria (e.g., does not have two prominent eyes), or if there are any other issues during image processing, the server returns an error message. The UI displays this error message to the user, indicating that the image could not be recognized or processed.

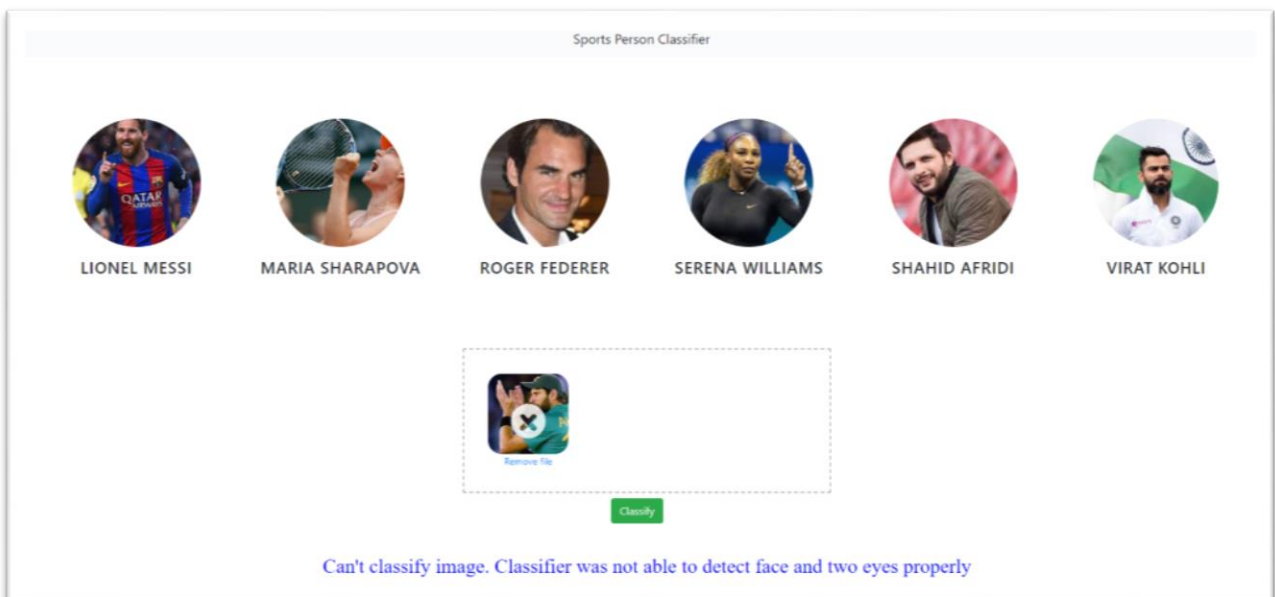


Figure 10: Error Handling

Results

Classification report

The classification report showcases the performance metrics of the Image Classifier system, providing insights into its accuracy, precision, recall, and F1-score across different categories. With an overall accuracy of 87%, the system demonstrates a commendable ability to correctly classify images into predefined classes.

Analysing the precision and recall values for each class reveals varying degrees of performance. Class 0 exhibits perfect precision but relatively lower recall, indicating that while the classifier correctly identifies instances of this class, it may miss some relevant instances. Conversely, Class 1 demonstrates high recall but slightly lower precision, implying that while the classifier identifies most instances of this class, it may also misclassify some instances from other classes as Class 1.

Classes 2 and 5 showcase a balance between precision and recall, indicating robust performance in both correctly identifying instances of the class and minimizing false positives. Class 4 stands out with perfect precision, recall, and F1-score, indicating flawless classification performance.

However, the system struggles with Class 3, demonstrating lower precision, recall, and F1-score compared to other classes. This suggests challenges in accurately classifying instances of this class, potentially due to class imbalance or inherent complexities in distinguishing features. The lower performance observed in Class 3 can be attributed to the quality of the images within this class. Images of lower quality often present challenges for image classification algorithms, as they may lack clear distinguishing features or contain artefacts that hinder accurate classification. In this case, the classifier struggles to correctly identify instances of Class 3 due to the inherent difficulties posed by low-quality images.

In summary, while the Image Classifier system demonstrates overall strong performance with an accuracy of 87%, there are variations in performance across different classes. Further optimization and fine-tuning may be required to address challenges in classifying certain classes and enhance the system's overall efficacy and robustness.

Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's performance for each class, revealing strong classification accuracy for classes 0, 1, 4, and 5, with minimal false positives and false negatives. However, Class 3 displays the weakest performance, characterized by a notably lower true positive count and a relatively high false negative count, which suggests challenges in correctly identifying instances of this class. This weaker performance aligns with the observation that low-quality images may have contributed to the model's difficulty in accurately classifying

Class 3 instances. Overall, the confusion matrix highlights areas where the model excels and areas where further refinement, particularly addressing image quality issues, may enhance classification accuracy.

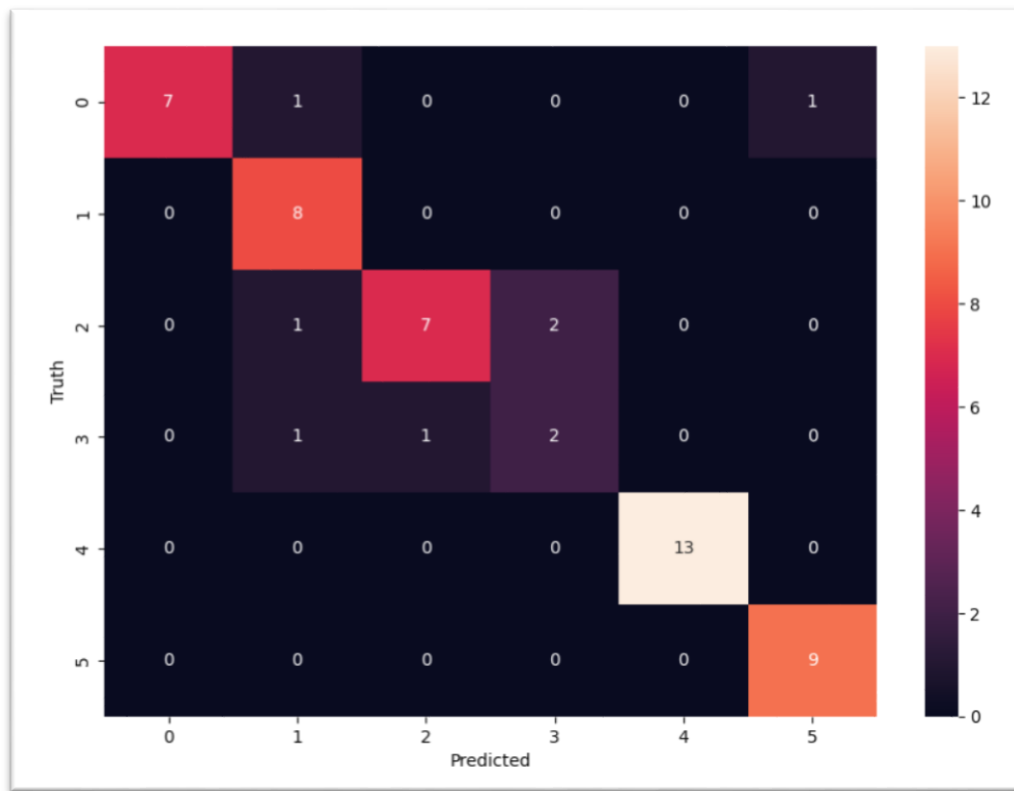


Figure 11: Confusion Matrix

Conclusion

The Image Classifier project integrates machine learning techniques with web development to provide a comprehensive solution for image classification tasks. Comprising the Model, Server, and User Interface components, this project offers a streamlined approach to accurately categorize images across various predefined classes. Leveraging Support Vector Machine (SVM) algorithms within the Model component, I aimed for simplicity and effectiveness, demonstrating that robust performance can be achieved without unnecessary complexity. However, challenges emerged, notably in accurately classifying certain images, particularly those of low quality, as evidenced by the weaker performance observed in Class 3. Despite this, the integration of multi-modal feature fusion techniques aimed to enhance classification accuracy, drawing from both raw pixel data and wavelet-transformed representations. Through experimentation and evaluation, methodology seeks to address these challenges, aiming to deliver accurate and efficient image classification capabilities to users across diverse domains. Furthermore, the exploration of various model

architectures, including Convolutional Neural Networks (CNNs), Random Forests, and Logistic Regression, underscores commitment to continually refining and optimizing the approach. With ongoing validation, optimization, and deployment efforts, I strive to ensure the effectiveness and reliability of the Image Classifier system, empowering users with intuitive image classification capabilities in real-world scenarios.

Future Work

Future work for this report involves several avenues for improvement and expansion. Firstly, to address the challenge of accurately classifying low-quality images, further research could focus on enhancing the model's robustness to variations in image quality through techniques such as data augmentation or incorporating image enhancement algorithms. Additionally, exploring state-of-the-art deep learning architectures like Convolutional Neural Networks (CNNs) may offer improved performance, especially for tasks involving complex image data.

Moreover, integrating multi-modal feature fusion techniques, as mentioned in the introduction, presents an opportunity to enhance classification accuracy by leveraging complementary information from both raw pixel data and wavelet-transformed representations. This approach could be further explored and refined to maximize the discriminative power of the classifier across diverse image datasets.

Furthermore, considering the potential applications mentioned, such as criminal data collection and patient record management, future work could involve adapting the image classifier for specific use cases and optimizing it for scalability and real-world deployment. This may entail incorporating additional functionalities, such as image retrieval and database integration, to support seamless data management and retrieval processes.

Additionally, exploring alternative model architectures beyond SVM, such as Random Forests or Logistic Regression, could provide insights into their comparative performance and suitability for different image classification tasks. Experimenting with ensemble methods or transfer learning techniques could also be beneficial in improving classification accuracy and generalization across various domains.

Lastly, incorporating user feedback and conducting usability studies to iteratively refine the user interface (UI) and enhance user experience would be crucial for ensuring widespread adoption and usability of the image classifier system. By addressing these areas of improvement and leveraging emerging technologies and methodologies, the image classifier can evolve into a versatile and reliable tool for a wide range of image classification tasks, fulfilling its potential across diverse domains and applications.

References

1. Carter, B. and Jain, S. (2021) *Overinterpretation reveals image classification model pathologies*.
https://proceedings.neurips.cc/paper_files/paper/2021/file/8217bb4e7fa0541e0f5e04fea764ab91-Paper.pdf Accessed.
2. Xin, M. and Wang, Y. (2019) Research on image classification model based on deep convolution neural network. *EURASIP Journal on Image and Video Processing* 2019 (1), 1–11.
3. Obaid, K.B., Zeebaree, S.R.M. and Ahmed, O.M. (2020) *Deep Learning Models Based on Image Classification: A Review*. <https://zenodo.org/record/4108433> Accessed 21 April 2024.
4. Chen, L., Li, S., Bai, Q., Yang, J., Jiang, S. and Miao, Y. (2021) Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Remote Sensing* 13 (22),
5. Lazebnik, S., Schmid, C. and Ponce, J. (2006) Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)* IEEE.
6. Bosch, A., Zisserman, A. and Munoz, X. (2007) *Image Classification using Random Forests and Ferns*.
<https://ieeexplore.ieee.org/abstract/document/4409066/references#references> Accessed 21 April 2024.
7. Jena, B., Saxena, S., Nayak, G.K., Saba, L., Sharma, N. and Suri, J.S. (2021) Artificial intelligence-based hybrid deep learning models for image classification: The first narrative review. *Computers in Biology and Medicine* 137 (2021), 104803.
8. A. G. Howard, G. Andrew et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications", *arXiv preprint arXiv: 1704.04861*, 2017.
9. Chandra, M.A. and Bedi, S.S. (2018) Survey on SVM and their application in image classification. *International Journal of Information Technology* 13 (5), 1–11.
10. mchaudh4 (2024) *Implementing Image classification using SVM*.
<https://stackoverflow.com/questions/76801704/implementing-image-classification-using-svm> Accessed 26 April 2024.
11. Pasolli, E., Melgani, F., Tuia, D., Pacifici, F. and Emery, W.J. (2019) *SVM Active Learning Approach for Image Classification Using Spatial Information*.
https://ieeexplore.ieee.org/abstract/document/6531640?casa_token=NbnBnpqIIOcAAAAA:BNuB613uTdfB8ivEeE8liSUdPx-bKEwjW9d-p5eD1s98-qcrM-nhNFt3qf8zrHkQ8nZHjFQ Accessed 26 April 2024.
12. Anon (n.d.) *Hyperspectral Image Classification—Traditional to Deep Models: A Survey for Future Prospects*. <https://ieeexplore.ieee.org/abstract/document/9645266> Accessed 27 April 2024.
- 13.