VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**NGUYEN DUY TUAN – 520K0232**

**TRINH BAO TOAN – 520K0332**

**CAT FACIAL COMPONENT DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**

**INFORMATION TECHNOLOGY PROJECT**

**COMPUTER SCIENCE**

**HO CHI MINH CITY, 2024**

VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**NGUYEN DUY TUAN – 520K0232**

**TRINH BAO TOAN – 520K0332**

**CAT FACIAL COMPONENT DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**

**INFORMATION TECHNOLOGY PROJECT**

**COMPUTER SCIENCE**

Advised by

**MSc. Nguyen Thanh An**

**HO CHI MINH CITY, 2024**

# ACKNOWLEDGEMENT

We would like to express my gratitude and thanks to our teacher Mr. An for his wonderful guidance throughout this project. We are also grateful to my parents for their continuous support to us throughout this experiment, to my friends for their help, and to all those who contributed directly or indirectly towards the completion of this school project. During this project, we acquired many valuable skills, and we hope that in the years to come, those skills will be put to good use.

*Ho Chi Minh city, 12th March 2024.*

*Author*

*(Signature and full name)*

***Tuan***

Nguyen Duy Tuan

***Toan***

Trinh Bao Toan

# DECLARATION OF AUTHORSHIP

We hereby declare that this is our own project and is guided by Mr. An Thanh Nguyen; The content research and results contained herein are central and have not been published in any form before. The data in the tables for analysis, comments and evaluation are collected by the main author from different sources, which are clearly stated in the reference section.

In addition, the project also uses some comments, assessments as well as data of other authors, other organizations with citations and annotated sources.

**If something wrong happens, We’ll take full responsibility for the content of my project.** Ton Duc Thang University is not related to the infringing rights, the copyrights that We give during the implementation process (if any).

*Ho Chi Minh city, 12th March 2024*

*Author*

*(Signature and full name)*

***Tuan***

Nguyen Duy Tuan

***Toan***

Trinh Bao Toan

**CAT FACIAL COMPONENT DETECTION USING CONVOLUTIONAL NEURAL NETWORKS**

# ABSTRACT

The project Cat Facial Component Detection (CFCD) is a burgeoning field within computer vision with wide-ranging applications in animal behavior analysis, biometrics, and veterinary medicine. In this report, we present a comprehensive study on the development and deployment of a robust CFCD system utilizing the MobileNetV2 architecture and Flask framework. Our methodology involves data collection, model training, and deployment stages. We fine-tuned MobileNetV2 using transfer learning on a curated dataset of cat images annotated with facial components. The trained model was encapsulated within a Flask API. Additionally, we developed an Android application that integrates with the Flask API, providing users with a user-friendly interface for capturing images and receiving CFCD results. Evaluation of the system demonstrates high accuracy and real-time performance on Android devices. Our work contributes to the advancement of computer vision techniques for animal behavior analysis and sets the stage for future research in this domain.

# TABLE OF CONTENT

[**ACKNOWLEDGEMENT 1**](#_lmbw3gffv4)

[**DECLARATION OF AUTHORSHIP 2**](#_wtat1lwvtqfr)

[**ABSTRACT 3**](#_b8y4q7k2drlt)

[**TABLE OF CONTENT 4**](#_lrh4kqisdq2n)

[**LIST OF FIGURES 6**](#_30j0zll)

[**LIST OF TABLES 7**](#_1fob9te)

[**ABBREVIATIONS 8**](#_3znysh7)

[**CHAPTER 1. INTRODUCTION 1**](#_2et92p0)

[2. The meanings of this project 1](#_uwqioncy6bdk)

[**CHAPTER 2. THEORETICAL BASIS 4**](#_1t3h5sf)

[2.1 Object detection in computer vision 4](#_4d34og8)

[2.1.1 Image Representation and Feature Extraction 4](#_gdu34kyid46n)

[2.1.2 Image Processing Techniques: 5](#_ma4k7hn0xbe0)

[2.1.3 Object Localization: 6](#_u82xy1vve0vh)

[2.1.4 Machine Learning Algorithms: 7](#_46rmk574289v)

[2.1.5 Deep Learning Architectures: 8](#_thn3x6atv3ie)

[2.1.6 Evaluation Metrics: 9](#_wt6iolkhj4vd)

[2.2 Theory of model 9](#_2s8eyo1)

[2.2.1 MobileNetV2 9](#_cmo3awlx8rem)

[2.2.2 Architecture of MobileNetV2 10](#_zfjj3ke17gt6)

[**CHAPTER 3. DESIGNING AND IMPLEMENTING 19**](#_17dp8vu)

[3.1 Data Collection and Preprocessing 19](#_3rdcrjn)

[3.1.1 Data collection 19](#_bk0hiqqaljo)

[3.1.2 Data Preprocessing 20](#_34dik9sgn2q6)

[3.2 Model Training 23](#_26in1rg)

[3.2.1 Data Normalization 23](#_28dvy7z928g1)

[3.2.2 Model Compilation and Training 24](#_prv1oqsth2d0)

[3.2.3 Model Evaluation and Validation 27](#_8t9rsdozfajf)

[**CHAPTER 4. SYSTEM APPLICATION 32**](#_lnxbz9)

[4.1 Back-end: 33](#_35nkun2)

[4.1.2 Module 33](#_3dyiqdgzat6s)

[4.1.3 Server 34](#_unr69lfeh1jx)

[4.2 Front-end 35](#_44sinio)

[4.2.1 HTML 35](#_dyzjmkyiangs)

[**CHAPTER 5. CONCLUSION 37**](#_z337ya)

[5.1 Conclusion 37](#_3j2qqm3)

[5.2 Limitations and Advantages 38](#_1y810tw)

[5.3 Future Development 39](#_oazxe67e5z5i)

[**REFERENCES 41**](#_4i7ojhp)

# LIST OF FIGURES

[Figure 2.1: Examples of ReLU transformations of low-dimensional manifolds embedded in higher-dimensional spaces. In these examples the initial spiral is embedded into an n-dimensional space using a random matrix T followed by ReLU, and then projected back to the 2D 23](#_Toc161856072)

[Figure 2.2: Evolution of separable convolution blocks. The diagonally hatched texture indicates layers that do not contain non-linearities. The last (lightly colored) layer indicates the beginning of the next block. Note: 2d and 2c are equivalent blocks when stacke 24](#_Toc161856073)

[Figure 2.3: The difference between residual block and inverted residual. Diagonally hatched layers do not use non-linearities. We use the thickness of each block to indicate its relative number of channels. 24](#_Toc161856074)

[Figure 2.4: Comparison of convolutional blocks for different architectures. 27](#_Toc161856077)

[Figure 3.1: The process of predicting landmarks 32](#_Toc161856078)

[Figure 3.2: Landmarks on a test image 32](#_Toc161856079)

[Figure 3.3: Landmarks and bounding boxes on a test image 33](#_Toc161856080)

[Figure 3.4: ReLU calculation formula 35](#_Toc161856081)

[Figure 3.5: MobileNetV2 building blocks 37](#_Toc161856082)

[Figure 3.6: MSE calculation formula 37](#_Toc161856083)

[Figure 3.7: Traing and Validation Loss of bounding boxes 38](#_Toc161856084)

[Figure 3.8: Traing and Validation Loss of landmarks 39](#_Toc161856085)

[Figure 3.9: List of example image for bouding boxes 40](#_Toc161856086)

[Figure 3.10: List of image for landmarks 41](#_Toc161856087)

[Figure 3.11: Sequence diagram of actions and interations between the user and the application 43](#_Toc161856088)

[Figure 4.1: Structure of the server 44](#_Toc161856089)

# LIST OF TABLES

[Table 2.1: Three layer of MobileNetV2 25](#_Toc161879055)

[Table 2.2: Inverted Residual and Linear Bottleneck 25](#_Toc161879056)

# ABBREVIATIONS

|  |  |
| --- | --- |
| CFCD | Cat Facial Component Detection |
| CNNs | Convolutional Neural Networks |
| CSS | Cascading Style Sheet |
| FCNs | Fully Convolutional Networks |
| GANs | Generative Adversarial Networks |
| HTML | HyperText Markup Language |
| IoU | Intersection over Union |
| JS | JavaScript |
| PCA | Principal Component Analysis |
| ReLU | Rectified Linear Unit |
| ResNet | Residual Networks |
| SSD | Single Shot Detectors |
| SVM | Support Vector Machine |
| YOLO | You Only Live Once |

# INTRODUCTION

The field of computer vision has witnessed remarkable breakthroughs in recent years, empowering machines to perceive, understand, and interact with the visual world akin to humans. One intriguing facet of this field is facial component detection, a task pivotal in various applications, including animal behavior analysis, biometrics, and augmented reality. In this report, we delve into the development and deployment of a sophisticated system for cat facial component detection leveraging MobileNetV2 architecture and Flask framework, culminating in the creation of an Android application.

1. **The limitations of this project**

The project has some disadvantages including:

Single-Facing Input Requirement: One notable disadvantage is the requirement for input pictures to feature the cat's face in a frontal orientation. This limitation arises from the model's training data and architecture, which may not effectively generalize to non-frontal poses. As a result, the system may struggle to accurately detect facial components in images where the cat's face is not prominently facing the camera.

The system can not recognize multi-cat faces

The system has a long time response (about 1-2s) because of the operation is held by server

## The meanings of this project

In the-state-of-the-art of computer vision and image analysis, developing solutions tailored to analyzing images of pets, particularly cats, holds significant meaning and practical applications:

**+ Understanding Animal Behavior:** Automated analysis of pet images enables researchers, veterinarians, and pet owners to gain insights into animal behavior patterns, expressions, and health indicators. For cats, understanding subtle facial expressions and body language can provide valuable clues about their mood, well-being, and potential health issues.

**+ Biometric Identification:** Identifying individual pets based on unique features such as facial markings or patterns aids in pet tracking, lost-and-found services, and personalized pet care. For instance, a solution capable of accurately recognizing a cat's facial features can assist in reuniting lost pets with their owners.

**+ Health Monitoring:** Analyzing pet images can facilitate early detection of health problems by identifying visual cues indicative of illness or discomfort. For cats, monitoring changes in facial expressions, body posture, or coat condition can help detect symptoms of diseases such as respiratory issues, dental problems, or skin disorders.

**+ Personalized Pet Care:** Tailoring pet care recommendations based on individual characteristics and behaviors enhances the well-being and quality of life for pets. Analyzing pet images can provide personalized dietary recommendations, behavioral training strategies, and enrichment activities tailored to each pet's unique needs and preferences.

+ **Research and Education:** Image analysis solutions contribute to scientific research and education by providing tools for studying animal behavior, cognition, and communication. Analyzing pet images helps researchers investigate evolutionary adaptations, social interactions, and environmental influences on pet behavior, contributing to our understanding of human-animal relationships.

**Solution for Automated Image Analysis on the Internet and Databases:**

Automated image analysis solutions deployed on the internet and integrated with databases offer several benefits and serve various purposes:

**+ Efficient Information Retrieval:** Automated image analysis algorithms enable users to search, filter, and retrieve relevant images from vast online repositories and databases efficiently. Users can specify search criteria based on visual features such as colors, shapes, textures, or semantic content, enhancing the accuracy and relevance of search results.

**+ Content Moderation and Filtering:** Image analysis solutions help enforce content moderation policies and filter inappropriate or harmful images on online platforms and social media networks. Automated algorithms can detect and flag images containing explicit content, violence, hate speech, or other violations of community guidelines, ensuring a safer and more inclusive online environment.

**+ Visual Search and Recommendation:** Integrating image analysis capabilities into e-commerce platforms and search engines enables users to perform visual searches and receive personalized product recommendations based on visual similarity. Users can upload images or input visual queries to discover visually similar items, facilitating product discovery, comparison, and purchase decisions.

**+ Data Mining and Insights Generation:** Automated image analysis algorithms extract valuable insights and patterns from large-scale image datasets, supporting data-driven decision-making and research in various domains. By analyzing visual content, researchers, marketers, and analysts can uncover trends, preferences, and consumer behaviors, informing product development, marketing strategies, and market forecasting.

**+ Enhanced Accessibility and Inclusivity:** Image analysis solutions contribute to enhancing accessibility and inclusivity by providing tools for image captioning, description, and interpretation. Automated algorithms generate textual descriptions or audio annotations for images, making visual content more accessible to individuals with visual impairments or cognitive disabilities, thus promoting digital inclusion and equal access to information.

## 

# THEORETICAL BASIS

## Object detection in computer vision

Object detection in computer vision is founded upon several theoretical principles and methodologies, drawing from various fields such as image processing, machine learning, and computer graphics.

Object detection aims to detect all instances of objects from known categories, such as humans, cars, or faces in an image or video. It's widely used in various applications including security and surveillance, image editing tools, machine inspection, autonomous vehicles, and many more.

### Image Representation and Feature Extraction

Understanding how images are represented and how features are extracted from them is fundamental in the realm of computer science, especially within areas such as computer vision, image processing, and machine learning. Let’s dive in to understand these concepts better.

In computer science, an image is represented as a matrix of pixel values. Pixels, short for picture elements, are the smallest units of an image that can be displayed and processed on a digital display device.

* Binary Images: The simplest form of image representation, where each pixel can only have one of two values, typically 0 (black) and 1 (white).
* Grayscale Images: In these images, each pixel carries a value representing shades of gray, where 0 is black, 255 is white, and the values in between represent various shades of gray.
* Color Images: Color images are typically represented in the RGB (Red, Green, Blue) color model, where each pixel consists of three values corresponding to the red, green, and blue components. The combination of these three colors at varying intensities can represent a wide spectrum of colors.

Feature extraction is the process of deriving informative and non-redundant characteristics from an image, which can be used for further analysis or processing, such as in image classification or recognition tasks.

* Edge Detection: Edges are significant local changes in intensity in an image. Edge detection algorithms like Sobel, Canny, or Prewitt are used to identify the boundaries of objects within images.
* Texture Analysis: Texture refers to the visual patterns that have properties of homogeneity that do not result from the presence of only a single color or intensity. Texture analysis can be used to classify regions in an image.
* Color Histograms: A color histogram represents the distribution of colors in an image. It counts how many times each color appears in the image. This is useful for image matching and classification.
* Shape Descriptors: Features that describe the geometric properties of an object in an image, such as area, perimeter, eccentricity, etc., are known as shape descriptors. These are crucial for object recognition and classification.

### Image Processing Techniques:

Image processing encompasses a broad spectrum of techniques designed to perform operations on images for their enhancement, information extraction, or conversion into a more desirable format. These techniques are pivotal across various domains, including computer vision, medical imaging, and video processing. One of the primary goals of image processing is image enhancement, which focuses on improving an image's visual appearance or adapting it for better analysis suitability. Techniques such as contrast adjustment, noise reduction, and sharpening fall under this category, aiming to make features within the image more distinguishable, reduce image noise while preserving details, and enhance object edges, respectively.

Another vital aspect of image processing is image restoration, which seeks to recover an image that has been degraded by known distortions, using mathematical or probabilistic models of image degradation. This includes deblurring techniques and inpainting, where the former removes blur effects and the latter reconstructs missing or damaged image parts. Color processing is also significant, involving manipulations like color space conversion and color balancing to correct color casts and achieve natural-looking images.

Morphological processing, dealing with the shape or structure of image features, employs operations such as erosion, dilation, opening, and closing to process images based on shapes, which can remove small objects or fill in small holes in an image. Edge detection and segmentation are crucial for identifying significant intensity transitions and partitioning an image into regions with similar attributes, using algorithms like Sobel, Canny, and methods like thresholding and clustering.

### Object Localization:

Object localization plays a crucial role in the field of computer vision, serving as a bridge between identifying the presence of objects in an image (object detection) and precisely pinpointing their locations. This task is typically represented by predicting bounding boxes around each object of interest within an image, which are defined by a set of coordinates that specify the location and size of these boxes. The process of object localization not only enhances the understanding of an image by providing spatial context but also enables a wide range of applications that rely on accurately locating objects within a visual scene.

The process of object localization involves several key steps, starting with feature extraction. In this initial phase, relevant features are extracted from the image, which could be achieved through traditional image processing techniques or more advanced approaches like Convolutional Neural Networks (CNNs). Following feature extraction, there's often a classification step that identifies the types of objects present in the image. Many modern techniques combine this step with the actual localization process, simultaneously predicting what the objects are and where they are located.

One of the pivotal aspects of object localization is the prediction of bounding boxes around detected objects. These boxes are typically defined by the coordinates of the top left corner, along with the width and height of the box, effectively encapsulating the object. Various techniques have been developed to improve the efficiency and accuracy of this process, including the sliding window method, region proposals, Fully Convolutional Networks (FCNs), and real-time detection and localization methods like Single Shot Detectors (SSD) and You Only Look Once (YOLO). Each of these approaches has its advantages and trade-offs, with some prioritizing accuracy while others focus on speed and real-time processing capabilities.

Object localization finds applications across a broad spectrum of fields and industries. In autonomous vehicles, it enables the identification and location of other vehicles, pedestrians, and obstacles, critical for safe navigation. Surveillance systems utilize object localization to monitor and alert authorities about the presence and movements of individuals or objects within their field of view. In the medical field, object localization aids in identifying anomalies such as tumors in medical scans, facilitating early diagnosis and treatment. Additionally, in the retail sector, it can be employed to analyze product arrangements on shelves, enhancing visual merchandising efforts.

### Machine Learning Algorithms:

Supervised Learning Algorithms are widely used in tasks where the model is trained on a labeled dataset. These labels indicate the correct output for each input image, and the algorithm learns to predict the labels from the input images. Common supervised learning algorithms include Support Vector Machines (SVMs), Convolutional Neural Networks (CNNs), and Random Forests. CNNs, in particular, have become synonymous with deep learning in computer vision due to their effectiveness in handling pixel data and their ability to learn hierarchical representations.

Unsupervised Learning Algorithms are utilized when there's no labeled data available. These algorithms attempt to learn patterns and structures from the data without any explicit instructions. Clustering and dimensionality reduction are common unsupervised learning tasks in computer vision, with algorithms such as K-means clustering and Principal Component Analysis (PCA) being frequently employed to group images based on similarities or to reduce the dimensionality of image data for analysis.

Semi-supervised Learning Algorithms fall between supervised and unsupervised learning, leveraging a small amount of labeled data alongside a larger set of unlabeled data. This approach is particularly useful in computer vision, where labeling large datasets can be prohibitively expensive or time-consuming. Semi-supervised learning algorithms can improve learning efficiency and accuracy by exploiting the unlabeled data during training.

Reinforcement Learning Algorithms are used in scenarios where an agent learns to make decisions by performing actions in an environment to achieve a goal. The agent receives rewards or penalties based on its actions, guiding it to learn the optimal strategy. In computer vision, reinforcement learning has been applied to tasks such as robotic vision, where the algorithm learns to navigate and interact with objects in its environment.

### Deep Learning Architectures:

Deep learning architectures have revolutionized the field of computer vision by enabling models to automatically learn hierarchical representations from raw data, significantly improving the accuracy and efficiency of visual data interpretation. These architectures, particularly Convolutional Neural Networks (CNNs), have become foundational to developing computer vision applications, ranging from image classification and object detection to more complex tasks like scene understanding and generative models.

Convolutional Neural Networks (CNNs) are at the forefront of deep learning in computer vision. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from visual data. This is achieved through the use of convolutional layers, pooling layers, and fully connected layers. Convolutional layers act as feature extractors that slide over input images, capturing spatial features such as edges and textures. Pooling layers reduce the dimensionality of the data, helping to make the representation more compact and allowing the network to focus on the most relevant features. Fully connected layers, typically found toward the end of a CNN, perform high-level reasoning based on the features extracted by the convolutional and pooling layers. The architecture of CNNs enables them to learn directly from pixel data, making them highly effective for tasks such as image classification, object detection, and semantic segmentation.

Following the success of CNNs, various specialized architectures have been developed to address specific challenges within computer vision. For instance, architectures like ResNet (Residual Networks) introduced the concept of skip connections to alleviate the vanishing gradient problem, allowing for the training of very deep networks by enabling the flow of gradients through the network. Similarly, architectures such as U-Net, designed for biomedical image segmentation, feature a symmetric expanding path that helps in precise localization.

Generative Adversarial Networks (GANs) represent another breakthrough in deep learning for computer vision, enabling the generation of new images that are indistinguishable from real images. GANs consist of two competing networks: a generator that produces images and a discriminator that evaluates them. The competition between these networks drives the improvement of generated image quality, leading to applications in image synthesis, style transfer, and data augmentation.

Attention mechanisms, initially developed for natural language processing tasks, have also been adapted for computer vision, leading to architectures like the Transformer. These mechanisms allow models to focus on relevant parts of the image, improving performance in tasks that require understanding of complex scenes or relationships between objects, such as image captioning and visual question answering.

### Evaluation Metrics:

Intersection over Union (IoU): IoU is a fundamental metric for evaluating the accuracy of object detection algorithms. It measures the overlap between predicted bounding boxes and ground truth annotations.

Precision, Recall, and F1-score: These metrics quantify the performance of object detection models in terms of their ability to correctly detect objects while minimizing false positives and false negatives.

## Theory of model

### MobileNetV2

MobileNetV2 is a significant advancement in the field of computer vision, particularly in the context of creating efficient models suitable for mobile and embedded devices. Developed to strike an optimal balance between computational efficiency and model performance, MobileNetV2 builds upon the ideas introduced in its predecessor, MobileNetV1, by incorporating a novel architectural concept known as the inverted residual structure with linear bottlenecks. This design enables the model to be lightweight without compromising on accuracy, making it ideal for applications where resources are limited.

At the core of MobileNetV2's efficiency is the inverted residual structure. Unlike traditional residual connections, which add the output of a convolutional block to its input, the inverted residuals first expand the input's dimensionality before applying a lightweight depthwise convolution and then compress the output back to a lower dimension. This approach reduces the computational burden while maintaining or even enhancing the representational capability of the network. The linear bottleneck, another key feature of MobileNetV2, refers to the use of linear rather than non-linear activation functions in the final layer of the residual block. This is based on the insight that nonlinearities can destroy too much information when applied in low-dimensional spaces, thereby reducing the network's effectiveness.

MobileNetV2's architecture allows it to excel in a wide range of computer vision tasks, including but not limited to image classification, object detection, and semantic segmentation. Its efficiency makes it particularly well-suited for real-time applications on mobile devices, such as augmented reality, face recognition, and real-time video analysis. The model's small size and high performance also make it an excellent choice for edge computing devices, where computational resources and power are limited.

The development of MobileNetV2 has had a profound impact on the deployment of deep learning models in real-world applications. By significantly reducing the computational cost of high-accuracy models, it has enabled advanced computer vision capabilities on a broad spectrum of devices, from high-end smartphones to low-power IoT devices. This democratization of deep learning technology has opened up new possibilities for application development, making sophisticated computer vision accessible to a wider range of developers and users.

### Architecture of MobileNetV2

**Depthwise Separable Convolutions**

Depthwise separable convolutions are a cornerstone in the MobileNetV2 architecture, providing an efficient method for reducing computational cost and model size while maintaining performance. This technique decomposes a standard convolution into two separate layers: a depthwise convolution and a pointwise convolution. This approach significantly reduces the complexity and computational resources required for the convolution operations, which are among the most computationally intensive parts of a deep neural network.

A standard convolution operation takes an input feature map of size *H*×*W*×*Din ,*

where *H* is the height, *W* is the width, and *Din* is the number of input channels, and applies *N* filters to produce an output feature map of size *H*′×*W*′×*N*, where

*N* is the number of output channels. The computational cost of this operation is proportional to *H*⋅*W*⋅*Din*​⋅*N*⋅*K*^2, where *K* is the size of the convolution kernel (assuming square kernels for simplicity).

In contrast, depthwise separable convolutions split this operation into two stages. First, a depthwise convolution applies a single filter per input channel, resulting in an intermediate feature map of the same dimensionality as the input (*H*×*W*×*Din*), but with filtered content. The computational cost of this stage is *H*⋅*W*⋅*Din*⋅*K^*2, significantly less than the standard convolution due to the absence of multiplication by the number of output channels *N.* Next, a pointwise convolution (a 1x1 convolution) is applied across the channels of the intermediate feature map to combine the filtered information and to expand (or contract) the channel dimension to the desired number of output channels *N.* The computational cost of this stage is

*H*⋅*W*⋅*Din*⋅*N*.

By decomposing the standard convolution into these two stages, the total computational cost of depthwise separable convolutions becomes *H*⋅*W*⋅*Din*⋅(*K^*2+*N*), which is significantly lower than that of standard convolutions, especially when *N* and *K* are large. This efficiency gain allows MobileNetV2 to deploy deeper and more complex models within the constraints of mobile and embedded devices.

MobileNetV2 enhances the depthwise separable convolution by introducing an inverted residual structure, where the input and output of the residual block are thin (low-dimensional), and the expansion occurs inside the block. This further optimizes the model's efficiency by reducing the dimensionality where the most computationally expensive operations occur.

**Inverted Residuals with Linear Bottlenecks**

MobileNetV2 introduces an innovative architectural feature known as Inverted Residuals with Linear Bottlenecks, which significantly enhances the efficiency and effectiveness of the network. This novel design marks a departure from traditional residual block structures by inverting the process of channel expansion and compression, coupled with the incorporation of linear bottlenecks to optimize the balance between performance and computational efficiency. This approach is central to MobileNetV2's ability to deliver high-performance deep learning models that are also highly efficient for deployment on mobile and embedded devices.

The architecture of an inverted residual block begins by expanding the input feature map to a higher dimensional space using a 1x1 convolution, contrary to the traditional method where the channel dimensions might first be compressed. This expansion increases the representational capacity of the network for the subsequent processing steps. Following this expansion, a depthwise separable convolution is applied, which performs lightweight filtering by applying a single filter per input channel. The process concludes with another 1x1 convolution that compresses the dimensions back to a lower-dimensional space, aligning with the concept of linear bottlenecks. This term, "linear bottleneck", refers to the avoidance of a non-linear activation function, such as ReLU, after the final 1x1 convolution, a strategy chosen to preserve the information content that might otherwise be lost through non-linearity, especially when reducing dimensions.

The inverted structure efficiently manages computational resources by performing the most demanding operations—such as depthwise separable convolutions—in a higher-dimensional space, where they are inherently more efficient. This efficiency does not come at the cost of performance, as the linear bottleneck ensures the crucial information is preserved through the compression phase of the process. The architecture provides a flexible framework that can be adjusted according to different computational budgets and performance requirements by modifying the expansion factor and the dimensions of the input and output channels.

In essence, the inverted residuals and linear bottlenecks framework is a cornerstone of MobileNetV2's design, enabling it to achieve an optimal balance between high accuracy in various computer vision tasks and the low computational cost necessary for effective deployment in mobile and embedded devices. This balance of performance and efficiency has led to the widespread adoption of MobileNetV2 in a variety of applications, from real-time image and video processing to complex machine learning tasks that require operating within the constraints of limited computational resources.

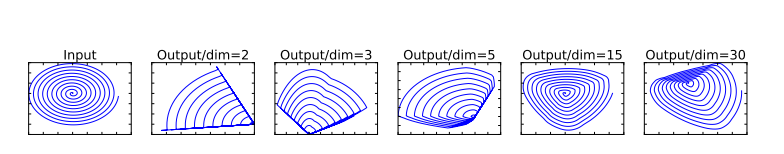


Figure 2.1: Examples of ReLU transformations of low-dimensional manifolds embedded in higher-dimensional spaces. In these examples the initial spiral is embedded into an n-dimensional space using a random matrix T followed by ReLU, and then projected back to the 2D

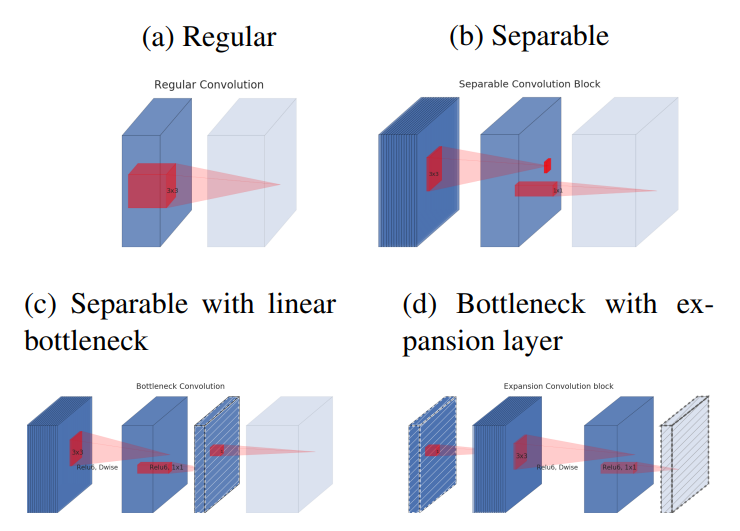


Figure 2.2: Evolution of separable convolution blocks. The diagonally hatched texture indicates layers that do not contain non-linearities. The last (lightly colored) layer indicates the beginning of the next block. Note: 2d and 2c are equivalent blocks when stacke

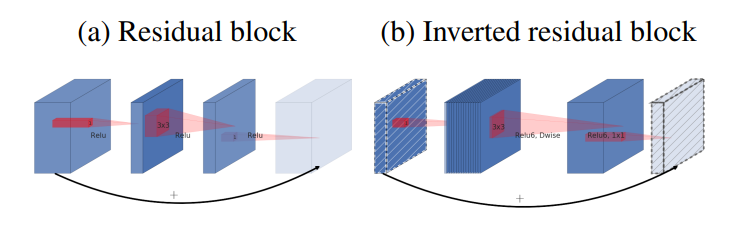


Figure 2.3: The difference between residual block and inverted residual. Diagonally hatched layers do not use non-linearities. We use the thickness of each block to indicate its relative number of channels.

Note how classical residuals connect the layers with a high number of channels, whereas the inverted residuals connect the bottlenecks. Best viewed in color.

**Model Architecture**

The MobileNetV2 architecture is designed to provide a highly efficient neural network model optimized for mobile and embedded device applications, maintaining a balance between performance and computational cost. This architecture builds upon the innovations introduced in its predecessor, MobileNetV1, by incorporating the Inverted Residuals with Linear Bottlenecks structure, which significantly enhances both efficiency and effectiveness. The core idea behind MobileNetV2 is to facilitate high-accuracy machine learning models that can be deployed on devices with limited computational power, such as smartphones, embedded systems, and IoT devices.

At the heart of MobileNetV2's architecture is a sequence of inverted residual blocks, each comprising an initial 1x1 convolution for expanding the input feature map's dimensionality, followed by a depthwise convolution for efficient spatial filtering, and concluding with another 1x1 convolution to compress the feature map and form a linear bottleneck. This structure is pivotal in reducing the computational burden while preserving the essential information across the network. Importantly, the use of linear bottlenecks avoids the loss of information that can occur with non-linear activations in low-dimensional spaces, enhancing the model's representational capacity.

The network begins with a standard convolutional layer, which is followed by 19 inverted residual blocks. These blocks are carefully arranged with varying channel sizes and expansion factors to optimize performance. The network concludes with a 1x1 convolution, a global average pooling layer, and a fully connected layer that outputs the final classification scores. MobileNetV2 also incorporates shortcut connections between the inverted residual blocks, similar to those in residual networks, which help mitigate the vanishing gradient problem and enhance the flow of information through the network.

One of the key features of MobileNetV2 is its adaptability to different computational budgets. By adjusting the width multiplier and the resolution multiplier, the model can be scaled to meet the specific constraints of an application, allowing for a flexible trade-off between accuracy and computational cost. This scalability, combined with the efficiency gains from the inverted residual blocks and linear bottlenecks, makes MobileNetV2 a versatile and powerful tool for a wide range of computer vision tasks, including image classification, object detection, and semantic segmentation, particularly in scenarios where computational resources are limited.

Table 2.1: Three layer of MobileNetV2

|  |  |  |
| --- | --- | --- |
| **Input** | **Operator** | **Output** |
| h x w x k | 1x1 conv2 , ReLU6 | h x w x (tk) |
| h x w x tk | 3x3 dwise s=s, ReLU6 | x x (tk) |
| x x tk | Linear 1x1 conv2d | x x k |

Table 2.2: Inverted Residual and Linear Bottleneck

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Operator** | **t** | **c** | **n** | **s** |
| 2242 x 3 | Conv2d |  | 32 | 1 | 2 |
| 1122 x 32 | Bottleneck | 1 | 16 | 1 | 1 |
| 1122 x 16 | Bottleneck | 6 | 24 | 2 | 2 |
| 562 x 24 | Bottleneck | 6 | 32 | 3 | 2 |
| 282 x 32 | Bottleneck | 6 | 64 | 4 | 2 |
| 142 x 64 | bottleneck | 6 | 96 | 3 | 1 |
| 142 x 96 | Bottleneck | 6 | 160 | 3 | 2 |
| 72 x 160 | Bottleneck | 6 | 320 | 1 | 1 |
| 72 x 320 | Cov2d 1x1 | - | 1280 | 1 | 1 |
| 72 x 1280 | Avgpool 7x7 | - | - | 1 | - |
| 1x1x1280 | Conv2d 1x1 | - | k | - | - |

The max number of channels/memory (in Kb) that needs to be materialized at each spatial resolution for different architectures. We assume 16-bit floats for activations. For ShuffleNet, we use 2x,g = 3 that matches the performance of MobileNetV1 and

MobileNetV2. For the first layer of MobileNetV2 and ShuffleNet we can employ the trick to reduce memory requirement. Even though ShuffleNet employs bottlenecks elsewhere, the non-bottleneck tensors still need to be materialized due to the presence of shortcuts between the non-bottleneck tensors.

**Memory efficient inference**

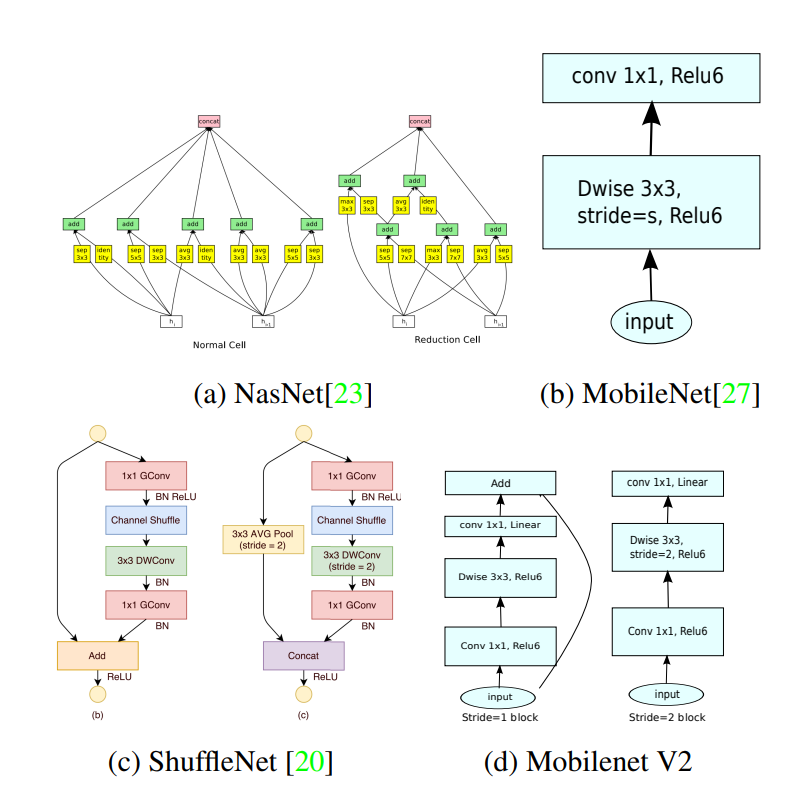


Figure 2.4: Comparison of convolutional blocks for different architectures.

MobileNetV2 is not only designed for high efficiency in computational terms but also optimizes for memory usage during inference, making it particularly suitable for deployment on mobile and embedded devices with limited memory resources. The architectural choices and innovations within MobileNetV2 contribute significantly to its memory-efficient inference capabilities.

The use of depth wise separable convolutions is a cornerstone of MobileNetV2's design that contributes to its memory efficiency. Unlike traditional convolutions that mix inputs across both spatial dimensions and depth (channels), depthwise separable convolutions split this process into two layers: a depthwise convolution that applies a single filter per input channel, and a pointwise convolution (1x1 convolution) that combines the outputs of the depthwise layer. This separation significantly reduces the number of parameters and the computational complexity, which in turn lowers the memory footprint both in terms of the model size (storage) and the activations (runtime memory).

Furthermore, the innovative structure of inverted residuals with linear bottlenecks in MobileNetV2 plays a crucial role in enhancing memory efficiency during inference. By expanding the feature space inside the block and then compressing it back before output, the architecture ensures that most of the computational work is done in a high-dimensional space where operations are inherently more efficient and require fewer memory resources for intermediate activations. The linear bottleneck, by avoiding non-linear activations in the final compression step, preserves information density and reduces the potential for information loss, which could otherwise necessitate a more complex (and memory-intensive) model to achieve similar accuracy.

MobileNetV2 also benefits from memory-efficient design through its adaptable architecture, which allows for the model to be scaled according to the available computational and memory resources. The width multiplier is a hyperparameter that scales the number of channels in each layer linearly, and the input resolution can be adjusted to manage the trade-off between accuracy and memory use. By reducing the width multiplier and input resolution, the memory footprint during inference can be significantly decreased, allowing for deployment on devices with very tight memory constraints without substantially compromising performance.

In practical deployment, memory-efficient inference of MobileNetV2 can be further enhanced by leveraging quantization and pruning techniques. Quantization reduces the precision of the model's parameters (and optionally the activations) from floating-point to fixed-point representation, which can drastically decrease the model size and speed up inference, often with minimal loss in accuracy. Pruning involves systematically removing less important parameters from the model, which not only reduces the model size but also decreases the memory required for storing activations during inference.

# DESIGNING AND IMPLEMENTING

## Data Collection and Preprocessing

The fundamental success of any computer vision task, a pivotal area within the realm of artificial intelligence that strives to endow machines with the capability to interpret and understand the visual world, is intricately tied to the quality and quantity of the data utilized during the training phase. In this comprehensive section, we meticulously delineate the process involved in the collection and preprocessing of the dataset, which serves as the cornerstone for training our sophisticated cat facial component detection model. This model is meticulously engineered to identify and analyze specific facial features of cats, thereby enabling a deeper understanding and interaction with these feline subjects through computational methods.

### Data collection

For our cat facial component detection model, we utilized the cat dataset available on Kaggle, consisting of over 9,000 cat images. Each image in the dataset is accompanied by annotations of the cat's head, specifying nine landmark points. These landmark points include two for the eyes, one for the mouth, and six for the ears:

+ **Number of Points:** This parameter represents the aggregate count of annotated points meticulously marked within our dataset, which, by default, is established at 9 points. This predefined number is critical as it forms the foundational basis for the precise identification and analysis of facial components in the dataset, facilitating the computational model's ability to learn and recognize these components with higher accuracy.

+ **Left Eye:** This entry specifies the precise coordinates of the left eye landmark, a pivotal feature in the realm of facial component detection. The left eye's coordinates are essential for the model to accurately identify and differentiate between various facial features, enabling a more nuanced understanding of feline facial structure and expressions.

+ **Right Eye**: Similar to the left eye, this entry delineates the exact coordinates of the right eye landmark. The accurate identification of both eyes' coordinates is crucial for maintaining the balance and symmetry in facial component detection, thereby enhancing the model's capability to accurately interpret and analyze feline facial expressions and features.

+ **Mouth**: This entry captures the coordinates of the mouth landmark, a key feature in understanding facial expressions and vocalization cues in felines. The precise mapping of the mouth coordinates aids in the detailed analysis of mouth movements and expressions, contributing significantly to the model's overall performance in detecting and recognizing various cat facial components.

+ **Left Ear(3 points):** This entry outlines the coordinates of three distinct points that collectively define the left ear, a feature of paramount importance in the accurate representation of the cat's ear structure. The triangulation of these three points allows for a more detailed and nuanced construction of the ear's shape and orientation, thereby enhancing the model's ability to recognize and interpret this specific facial feature.

+ **Right Ear(3 points)**: Analogous to the left ear, this entry provides the coordinates for three points that define the right ear's structure. The precise delineation of these points is essential for constructing a symmetrical and accurate representation of the cat's ear, which is instrumental in achieving a balanced and comprehensive understanding of the cat's facial anatomy within the computational model.

### Data Preprocessing

Prior to initiating the model training, we undertook a comprehensive preprocessing routine on our dataset. This meticulous preparation was crucial to ensure that the data was fully compatible with our chosen model architecture and to optimize the overall performance of the model. Our preprocessing pipeline was meticulously designed to encompass several pivotal steps, each tailored to refine the dataset for optimal model training outcomes.

**Bounding Box Extraction:**

The initial step in our preprocessing pipeline involved the extraction of bounding boxes around the cat's head, utilizing the provided landmark annotations. This process was carried out by calculating the minimum and maximum coordinates across the landmark points, thereby delineating a bounding box that effectively encompassed the entire head region. The rationale behind extracting these bounding boxes was to establish spatial constraints, which are instrumental in training the model to accurately detect facial components within the specified region.

**Landmark Relocation and Normalization**

Following the extraction of bounding boxes, we proceeded to relocate and normalize the landmark points in relation to the bounding boxes. This entailed translating the landmark coordinates to align with the bounding box's top-left corner, followed by scaling these coordinates to fit within a standardized image dimension of 224x224 pixels. The normalization of landmark positions is a critical step, as it standardizes the input data, enabling the model to learn and identify robust features effectively, irrespective of variations in the sizes and orientations of the cats' heads.

**Image Resizing and Augmentation**

An integral part of our preprocessing involved resizing all images to a uniform resolution of 224x224 pixels. This step was essential not only to standardize the input size for the model but also to preserve the spatial relationships between the facial components across different samples, thereby facilitating a more efficient model training process. Moreover, to bolster the diversity of our dataset and enhance the model’s ability to generalize, we employed various data augmentation techniques, including random rotations, flips, and translations. These augmentation strategies are pivotal in creating a robust dataset that mimics a wide range of real-world variations.

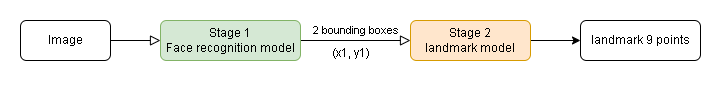


Figure 3.1: The process of predicting landmarks

**Visualization:**

Throughout the entire preprocessing pipeline, we consistently employed visualization techniques to scrutinize the processed images and landmark annotations. This practice was crucial for verifying the accuracy of our transformations and maintaining the integrity of the data. Visualization served as a quality control measure, allowing us to detect and rectify any anomalies or inconsistencies within the dataset, thereby refining our preprocessing steps in an iterative manner.

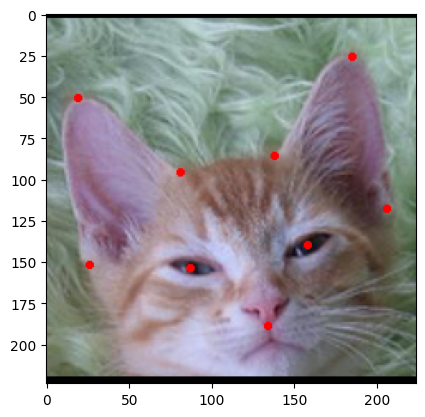
****

Figure 3.2: Landmarks on a test image

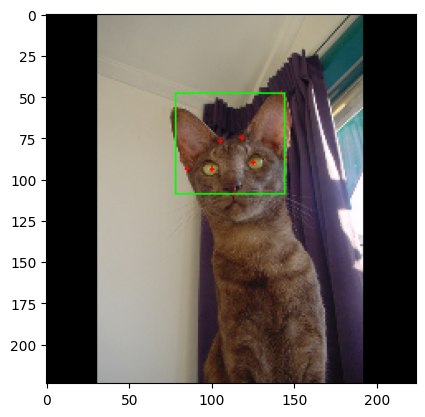
****

Figure 3.3 Landmarks and bounding boxes on a test image

## Model Training

### Data Normalization

Before the training phase of the model can commence, a critical and foundational step involves the meticulous preprocessing and normalization of the data. This stage is paramount for setting a solid groundwork that significantly influences the effectiveness and efficiency of the model's learning process. The dataset in question consists of an assortment of cat images, each accompanied by detailed annotations that mark key facial landmarks along with bounding box coordinates. Specifically, these annotations delineate nine critical points on each cat's face, capturing the locations of two eyes, one mouth, and six ear points (typically, it's important to clarify that cats have two ears, but the annotations might refer to multiple points per ear to define the ear's shape or outline).

The process of data normalization is a key aspect of this preparatory phase. Normalization refers to the transformation of data to a common scale without distorting differences in the ranges of values or losing information. For the cat images and their corresponding annotations, this step involves converting the landmark annotations and bounding box coordinates into a floating-point format. This conversion facilitates mathematical operations and computational processing. Furthermore, each of these values is scaled to fit within a normalized range of [0, 1]. This scaling ensures that the model treats all input data uniformly, regardless of the original scale or dimensions of the images. By doing so, the model can more easily detect patterns and features across different images, thereby enhancing its capability to generalize from the training data to new, unseen images.

Normalization aids in achieving uniformity across the dataset and significantly contributes to the convergence of the model during the training process. Convergence refers to the point at which the model's learning reaches an optimal state, where further training does not significantly improve its performance. Normalized data helps in smoothing the learning curve and reducing the number of training epochs required to reach convergence, thereby making the training process more efficient and less resource-intensive.

Moreover, the uniform scaling of landmark annotations and bounding box coordinates to the [0, 1] range facilitates the model's adaptability and flexibility in handling images of varying sizes and resolutions. Since the model learns to detect and interpret normalized coordinates, it can accurately apply this knowledge to any image, regardless of its original scale. This adaptability is particularly beneficial in real-world applications, where the model may encounter images from different sources and contexts.

In addition to normalization, this preprocessing stage may also involve data augmentation, where the existing dataset is artificially expanded by applying various transformations to the images, such as rotations, scaling, translations, and flips. Data augmentation enhances the diversity of the training dataset, which further strengthens the model's generalization capabilities by exposing it to a wider range of variations within the data.

### Model Compilation and Training

The MobileNetV2 model is configured with pre-trained weights obtained from the ImageNet dataset, omitting the fully connected layers *(include\_top=False)*. The extracted features are then passed through a global average pooling layer followed by two dense layers with Rectified Linear Unit (ReLU) activation functions. This architecture is tailored to extract relevant features from the input images and facilitate accurate bounding box predictions. ReLu is defined as:



Figure 3.4ReLU calculation formula

In the development of the model, the architecture begins with an input layer designed to process images of a specific size, namely (224, 224, 3). This dimensionality ensures that the input images are standardized, allowing for consistent processing across all data the model encounters. The choice of these dimensions aligns with common practices in image recognition tasks, providing a balance between detail retention and computational efficiency.

Following the input layer, MobileNetV2 serves as the backbone architecture of the model. This choice is strategic, as MobileNetV2 is renowned for its efficiency and effectiveness in mobile and embedded vision applications. It's initialized with pre-trained weights from ImageNet, a large visual database often used for image recognition software training. By employing this pre-trained model and removing the top layer, the model leverages learned features from a vast array of images, boosting its ability to recognize and analyze new images. This approach significantly accelerates the training process and enhances the model's performance by utilizing knowledge gained from previously learned patterns.

Subsequent to the MobileNetV2 architecture, a Global Average Pooling (GAP) 2D layer is introduced. The GAP layer plays a crucial role in reducing the dimensionality of the feature maps generated by the previous layers to a vector. It accomplishes this by taking the average of each feature map, effectively summarizing the spatial information. This step is crucial for transitioning from the high-dimensional output of convolutional layers to a more compact form suitable for classification tasks, reducing the model's complexity without sacrificing essential information.

To further process the information, two dense layers follow the GAP layer. These layers, containing 128 and 64 units respectively, utilize the ReLU (Rectified Linear Unit) activation function. The inclusion of dense layers with ReLU activation introduces non-linearity to the model, enabling it to learn more complex patterns in the data. The choice of ReLU is motivated by its effectiveness in avoiding the vanishing gradient problem, thus facilitating the training of deep neural networks.

The architecture culminates in an output layer designed to make the final prediction. This layer employs a linear activation function, making it suitable for a wide range of applications, including regression and classification tasks. The linear activation ensures that the output can be directly interpreted as a prediction score, which can be further processed depending on the specific requirements of the task at hand.

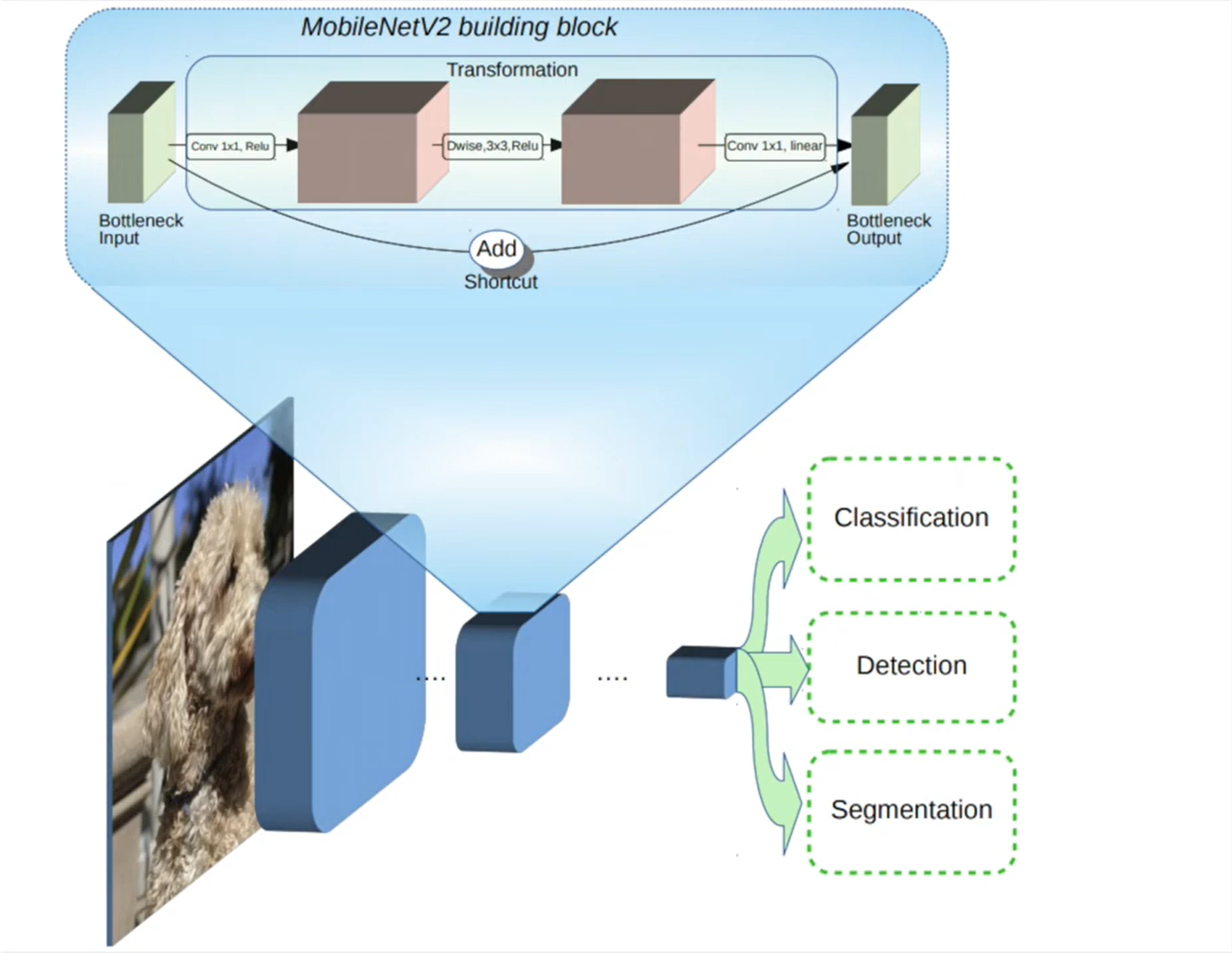


Figure 3.5: MobileNetV2 building blocks

Once the model architecture is defined, it is compiled to specify the optimization algorithm, loss function, and evaluation metrics. In this case, the Adam optimizer is selected due to its efficiency and adaptability. The Mean Squared Error (MSE) loss function is utilized, given its suitability for regression tasks such as bounding box prediction. The MSE of the predictor is computed as:

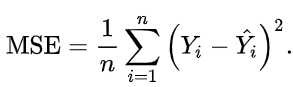


Figure 3.6: MSE calculation formula

The model is then trained using the compiled configuration and the training data. The training process involves iteratively feeding batches of images and their corresponding ground truth bounding box coordinates into the model. Over the course of 50 epochs, the model learns to minimize the discrepancy between its predictions and the true bounding box coordinates.

### Model Evaluation and Validation

Throughout the training process, the model's performance is continuously monitored on the validation set. At the end of each epoch, the model's loss on both the training and validation sets is computed and compared. This enables the detection of overfitting or underfitting phenomena and allows for necessary adjustments to the model architecture or training procedure. Below is the Training and Validation Loss for bounding boxes and landmarks respectively:

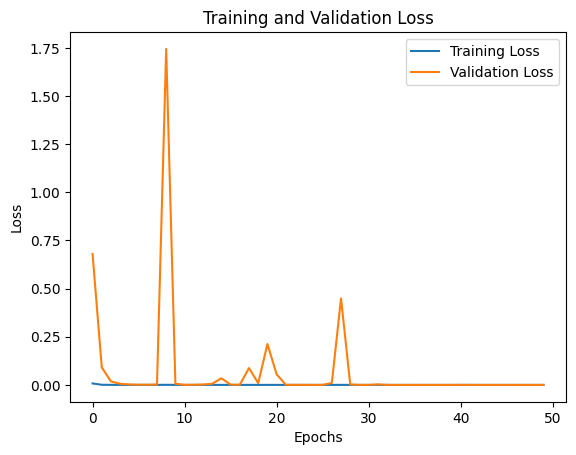
****

Figure 3.7: Traing and Validation Loss of bounding boxes

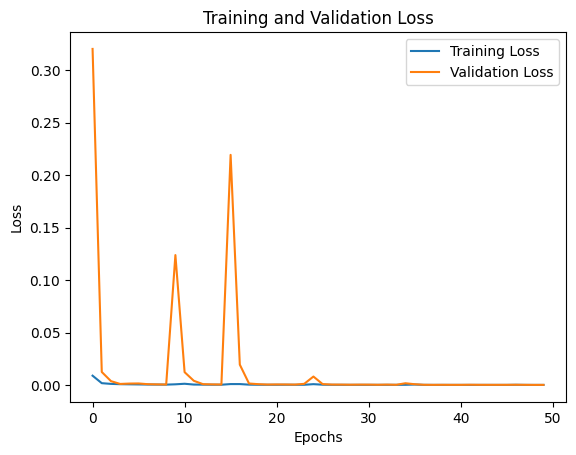
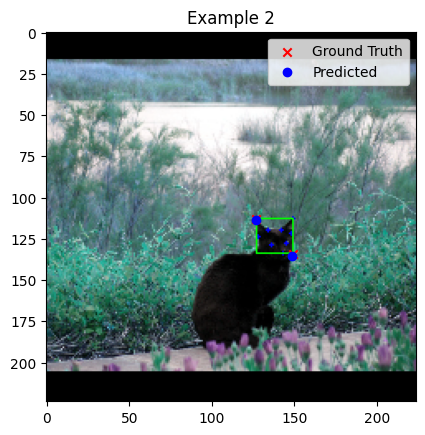
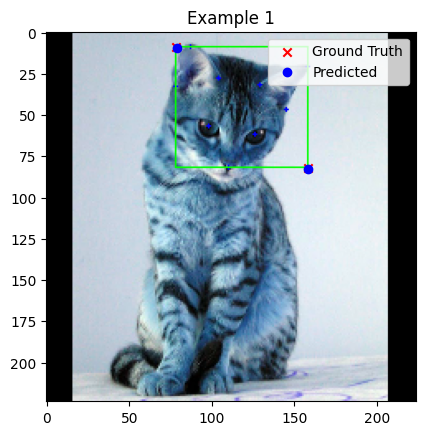
****

Figure 3.8: Traing and Validation Loss of landmarks

To assess the model's efficacy visually, sample images from the testing set are randomly selected, and their corresponding ground truth and predicted bounding box coordinates are overlaid on the images. This visualization aids in understanding the model's performance qualitatively and provides insights into areas of improvement.

Example for the bounding boxes:



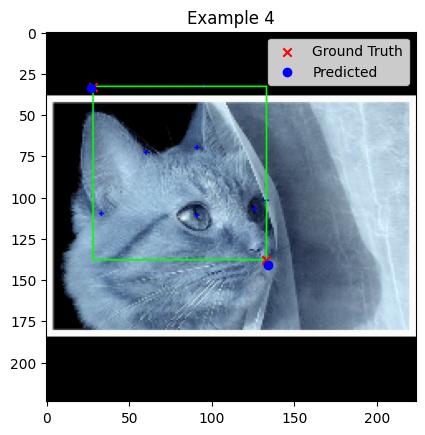
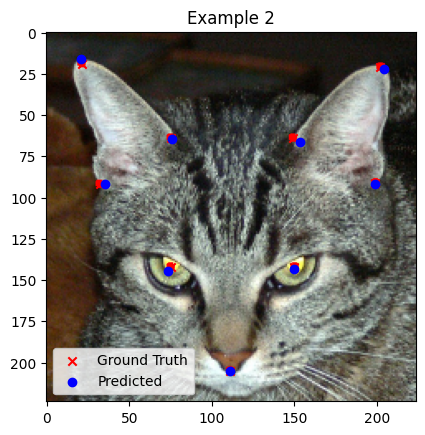
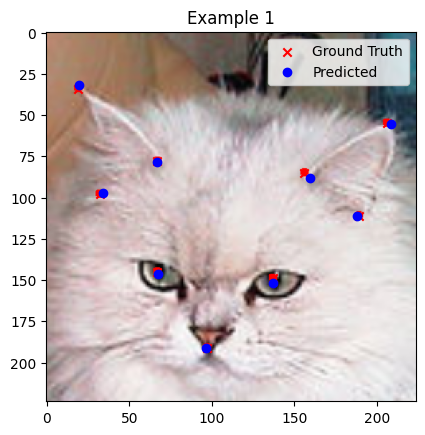


Figure 3.9 List of example images for bouding boxes

Example for the landmarks:



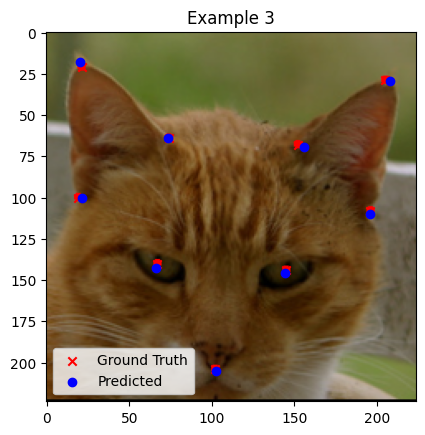


Figure 3.10: List of image for landmarks

### 

# SYSTEM APPLICATION

Developing an application system that leverages the MobileNetV2 architecture for recognizing cat faces, including their bounding boxes and landmarks, represents a fascinating intersection of deep learning and web development. This system can serve a wide range of practical purposes, from enhancing pet-related applications to contributing to research on animal behavior and welfare. Here, we explore how such a system can be constructed, focusing on backend development with Flask and frontend visualization using HTML, JavaScript (JS), and CSS.

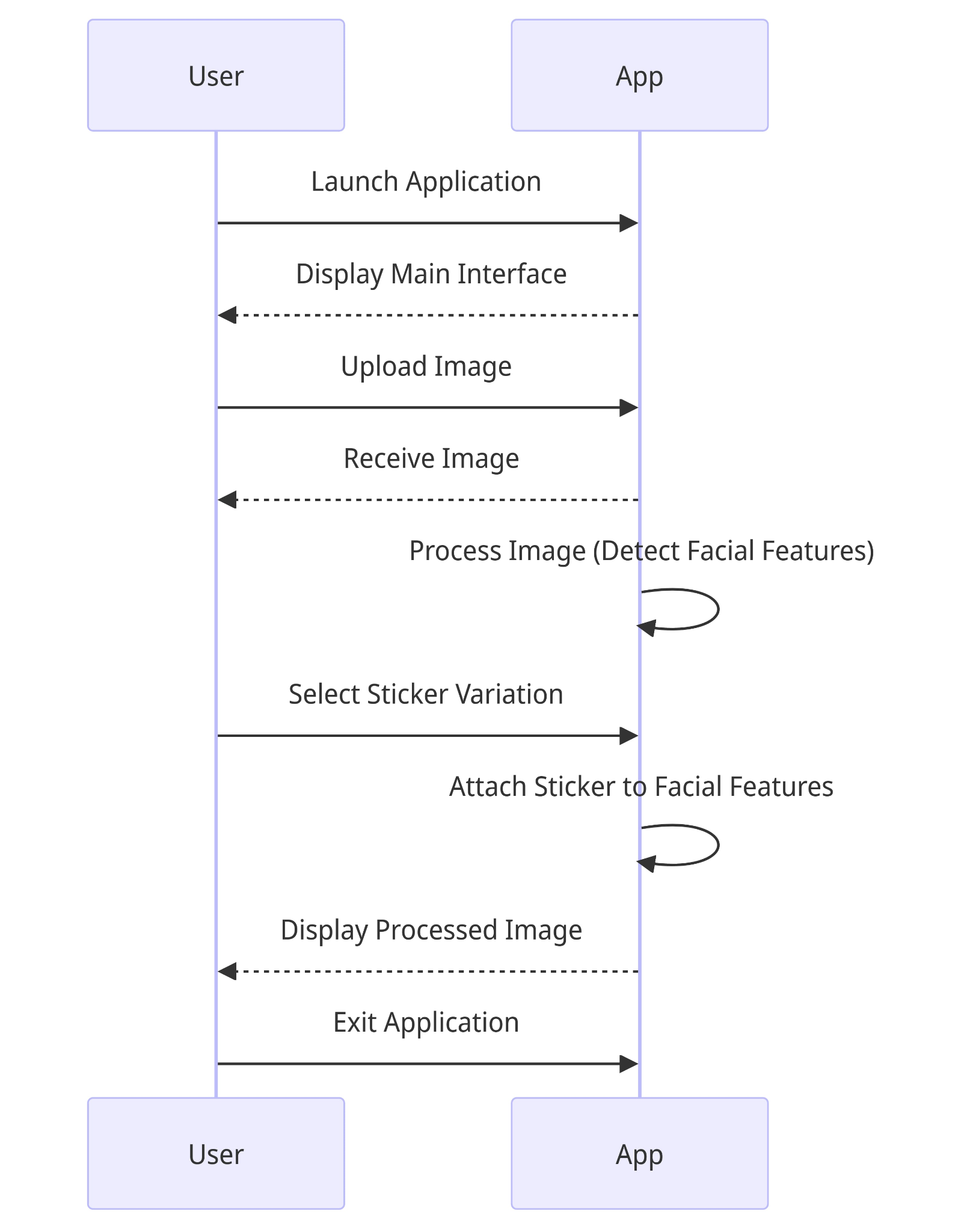


Figure 4.1 Sequence diagram of actions and interactions between the user and the application

## Back-end:

* + 1. **Structure of server**

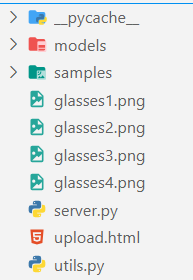


Figure 4.2: Structure of the server

### Module

Initially, At the core, the module imports essential libraries such as TensorFlow, OpenCV (cv2), and NumPy, which are critical for handling deep learning models, image processing, and numerical calculations, respectively. TensorFlow is used to load pre-trained deep learning models that predict bounding boxes (bbs) and facial landmarks (lmks) from images. These models are encapsulated in functions \_load\_model\_lmks() and \_load\_model\_bbs(), which load the models from specified paths and are ready to be used for inference.

The resize\_img function is designed to resize images while maintaining their aspect ratio. This is particularly important for preparing images before passing them to deep learning models, which often require input images of a fixed size. The function calculates a new size for the image based on the desired target size (img\_size), resizes the image, and adds padding to ensure that the final image meets the required dimensions. This preprocessing step is crucial for maintaining the integrity of the image's content while conforming to the model's input specifications.

The overlay\_transparent function is a more advanced image manipulation method that overlays a transparent image (such as glasses) onto a background image at a specified location and with an optional size. This function is particularly useful in applications such as virtual try-on systems, where accessories need to be accurately positioned on the user's face. The function handles the complexities of blending images with transparency, ensuring that the overlay is seamlessly integrated with the background image.

Finally, the angle\_between function calculates the angle between two points, which can be useful for adjusting the orientation of the overlaid image based on the angle of facial features detected by the landmark prediction model. This is an essential step for ensuring that accessories like glasses align correctly with the user's face, enhancing the realism of the overlay.

### Server

Flask-based web application designed to incorporate advanced image manipulation capabilities, specifically to overlay glasses on faces within uploaded images. This application leverages several key Python libraries, including Flask for web service creation, OpenCV and Pillow for image processing, and TensorFlow for leveraging deep learning models. The core functionality is split across multiple endpoints, each serving a distinct purpose in the image manipulation process.

The application features a simple "Hello World" endpoint, an image upload mechanism, a prediction endpoint for overlaying glasses on faces, and a combined upload and predict feature for a streamlined user experience. Additionally, it offers a basic HTML interface for user interaction with the service. The prediction logic utilizes computer vision techniques to detect faces and identify facial landmarks, which are crucial for accurately positioning the glasses overlay on the images. This is achieved through pre-trained deep learning models that predict bounding boxes around faces and specific facial landmarks.

The application's architecture demonstrates a practical application of integrating Flask with powerful image processing and deep learning libraries to create a highly interactive and functional service. However, there are areas for improvement, such as adding robust validation and error handling for image uploads, optimizing performance to handle potentially slow processing times due to complex models and image manipulations, enhancing the user interface for a better user experience, and considering scalability options for handling higher traffic volumes. Overall, this Flask application exemplifies how web development can be combined with computer vision and deep learning to produce innovative and interactive web services.

## Front-end

### HTML

This HTML document is designed as a front-end interface for a web application focused on detecting cat facial components. It is meticulously crafted, incorporating styling, structure, and basic interactivity scripts. These elements work together to manage image uploads, display user-selected images, and facilitate server interactions for image processing. The document's structure is primarily built using <div> elements, which are then enhanced with CSS to achieve a modern and clean user interface. A significant feature, the .container class, centralizes the main content area with a design that adapts responsively to different screen sizes.

The introduction of the application is marked by the #logo section, which prominently features an image and a title, strategically positioned at the top-left corner of the container for immediate visibility. The interface enhances user interaction through an ingeniously hidden file input field and a visually appealing label that serves as the upload button, thereby elevating the overall aesthetic. Furthermore, the application offers users the ability to select from various processed image variations through a set of radio buttons, each accompanied by an image. This feature allows users to choose their preferred style of cat facial component overlays.

A notable aspect of the interface is the implementation of a loading spinner, crafted with CSS animations. This spinner plays a crucial role in providing users with visual feedback during asynchronous operations, such as when the application is processing image requests. On the functionality front, JavaScript adds a layer of interactivity to the application. Functions like displaySelectedImage take charge of showcasing the user-selected image within the #originalImageContainer by creating an Object URL from the file input. Another critical function, submitFormAndDisplayProcessedImage, overrides the default form submission behavior to manage the image upload and processing request asynchronously via the Fetch API.

Despite the robust framework, the document is not without its flaws. A minor oversight is observed in the submitFormAndDisplayProcessedImage function, where it incorrectly targets #originalImageContainer instead of #processedImageContainer for displaying the processed image. Additionally, the document redundantly includes two identical blocks for #processedImageContainer and .loading, a detail that could be streamlined for efficiency. To enhance the user experience further, integrating feedback or error messages into the UI could provide users with clearer processing status updates or upload failure notifications. Moreover, incorporating alt attributes to the radio button images would significantly improve accessibility, ensuring that screen readers can effectively communicate the variations to visually impaired users.

From a security standpoint, it is imperative that the server handling the image processing enforces stringent data sanitization and validation measures. This precaution is vital to safeguard against common web vulnerabilities, particularly those related to file uploads. In summary, this code lays a solid foundation for a web-based image processing application. Nonetheless, there exists potential for enhancements in areas such as error handling, accessibility, and security practices, which would further refine the application's functionality and user experience.

#### 

# CONCLUSION

## Conclusion

The development of the Cat Facial Component Detection web application has been a remarkable success, achieving its primary objective of providing users with a powerful tool for detecting and visualizing facial components on cat images. This accomplishment was made possible through the integration of cutting-edge technologies and the implementation of key features that deliver an exceptional user experience.

One of the notable achievements of the application is its seamless image upload functionality. Users can effortlessly upload images of cats directly from their local devices, streamlining the process of inputting data for facial component detection. This feature not only enhances accessibility but also ensures a smooth and intuitive user experience, fostering widespread adoption and engagement with the application.

Moreover, the application offers users a diverse range of glasses variations to overlay onto the uploaded cat images. This feature adds an element of customization and creativity, allowing users to express their individuality and artistic flair. By enabling users to personalize the visual representation of their beloved feline companions, the application fosters a deeper connection and enjoyment for its users.

Underpinning the application's core functionality is the real-time processing capability, which ensures prompt feedback and visualization of the detected facial components. Leveraging advanced backend processing power, the web application performs real-time analysis and processing of the uploaded images, eliminating significant delays or interruptions. This real-time responsiveness enhances the overall user experience, providing a seamless and engaging interaction with the application.

Perhaps the most significant achievement of the Cat Facial Component Detection web application is the integration of the powerful MobileNetV2 deep learning model for facial component detection. This sophisticated technology enables the application to achieve remarkable accuracy and reliability in identifying and visualizing various facial features on cat images. By harnessing the capabilities of this advanced deep learning model, the application delivers precise and efficient detection of facial components, ensuring that users receive accurate and meaningful results.

## Limitations and Advantages

While the project has successfully met its main goals, it's important to acknowledge its limitations alongside its achievements. One significant limitation is the model's capability to detect only a single cat per frame. This constraint could limit the application's utility in scenarios where images contain multiple cats, potentially restricting its broader applicability. Additionally, the model's performance is subject to variation based on the facial orientation of the cat. It exhibits robust performance for frontal faces but may underperform with side-facing or oblique angles, which limits its ability to accurately detect and analyze facial components across diverse cat poses. Moreover, the model operates under the assumption that a cat is present within the image, which means it lacks the capability to discern scenarios where no cat exists. This could lead to false positives or inaccurate results in cat-absent images, affecting the application's reliability.

On the achievements front, the project has made notable strides. The model demonstrates powerful detection capabilities for frontal-facing cat images, ensuring precise identification and visualization of key facial features. This capability is crucial for detailed analysis and interpretation of cat expressions and characteristics. Furthermore, the application has enhanced its user experience by integrating various glasses variations, adding a fun, creative element that enhances its appeal and usability. This feature makes the application more engaging for a wide audience. Another significant achievement is the application's real-time processing capability, enabled by advanced backend processing. This ensures users receive prompt feedback and interaction, facilitating seamless and efficient usage without noticeable delays or interruptions.

In summary, despite its limitations—such as single cat detection per frame, performance variation based on facial orientation, and inability to detect the absence of a cat—the project has made commendable achievements. These include powerful frontal face detection, enhanced visualization through creative features, and real-time processing capabilities. These strengths and weaknesses reflect the project's current state and provide a roadmap for future improvements and expansions.

## Future Development

As the primary objective of the project has been met, there are numerous opportunities for future development and enhancement that could significantly elevate the application's functionality and user experience. One such enhancement involves introducing the ability to detect specific facial features, such as the head and nose, enabling users to attach customizable stickers or accessories either in real-time or during post-processing. This feature would not only enhance user engagement by allowing pet owners to personalize images of their cats with decorative elements but also increase creativity through sticker customization options like resizing, rotating, and choosing from a wide range of designs.

In addition to personalization features, expanding the model's capabilities to detect and analyze multiple cats within a single frame would significantly broaden the application's applicability. This improvement is particularly relevant in environments with multiple cats, such as pet shelters, veterinary clinics, or households with more than one pet, making the application more versatile and valuable in a wider range of scenarios.

Further, enhancing the model's performance for side-facing or oblique cat images is another critical area for development. Investing in research and the implementation of advanced algorithms or alternative neural network architectures tailored to diverse facial orientations could lead to substantial improvements in accuracy and reliability for detecting and analyzing cats in various poses.

Lastly, improving the model's generalization capabilities by training it on a more diverse dataset that includes a wide range of cat breeds, ages, and fur colors is essential. This approach would ensure the model's ability to accurately detect and analyze facial components across a broader spectrum of cat demographics, resulting in more robust and inclusive outcomes.

These future development avenues-encompassing sticker attachment and customization, multi-cat detection, improved performance for side faces, and enhanced generalization—outline a roadmap that could significantly enhance the application's functionality, making it more engaging, versatile, and accurate.

# REFERENCES

[1] King, D. (n.d.). *Hipsterize your dog with Deep Learning*. dlib C++ Library. https://blog.dlib.net/2016/10/hipsterize-your-dog-with-deep-learning.html

[2] Kairess. (2018b, October 23). *Cat Hipsterizer*. Kaggle. https://www.kaggle.com/code/kairess/cat-hipsterizer

[3] Khanh. (n.d.). *Khoa Học dữ liệu*. Khanh’s blog. https://phamdinhkhanh.github.io/2020/03/23/FlaskRestAPI.html

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” Accessed: Mar. 21, 2024. [Online]. Available: https://arxiv.org/pdf/1801.04381.pdf?fbclid=IwAR1RFQ5YQCMRdpj8yu621DFrluhGM-X7-xt-O85LC\_O4BlgmBrhzMRlOpVw

‌