Faculty of Engineering & Technology			
Ramaiah University of Applied Sciences			
Department	Computer Science and Engineering		
Programme	B. Tech. in CSE		
Batch	2022	Course Start Date	03-03-2025
Course Code	**20AIC313A**		
Course Title	Computer Vision		
Course Leader(s)	Ms.Anitha K R		

Assignment Assessment			
Reg.No.	22ETCS002028	Name of the Student	**Chinmay K Shetty**

Report on Assignment				
	Marking Scheme		Marks	
			Max Marks	Examiner Marks
	1.1	Write a Python code using OpenCV to perform the following i) Convert RGB Image to Grey scale ii) Change the resolution to 512 X 512 iii) Increase the contrast of the image using histogram equalization iv) Display the histogram of the image before and after histogram equalization. v) Apply Median filter on the image	10	
	1.2	Write a detailed note on Deep Learning for Computer Vision	10	
	1.3	Discuss the challenges in implementing Deep Learning for Computer Vision to solve the real word problems.	5	
		Max Marks	25	
Signature of Examiner				

Instructions to students:

1. The assignment has to be neatly word processed as per the prescribed format.
2. The maximum number of pages should be restricted to 10 pages.
3. Use only SI units
4. **Submission Date: 13/06/2025**
5. **Submission after the due date is not permitted.**
6. Method of evaluation as per the submission and marking scheme
7. At the end, you are required to comment on -
   Benefits you have derived by solving this assignment
8. IMPORTANT: It is essential that all the sources used in preparation of the assignment must be suitably referenced in the text.

1.1 **Write a Python code using OpenCV to perform the following**
i) Convert RGB Image to Grey scale
ii) Change the resolution to 512 X 512
iii) Increase the contrast of the image using histogram equalization
iv) Display the histogram of the image before and after histogram equalization.
v) Apply Median filter on the image

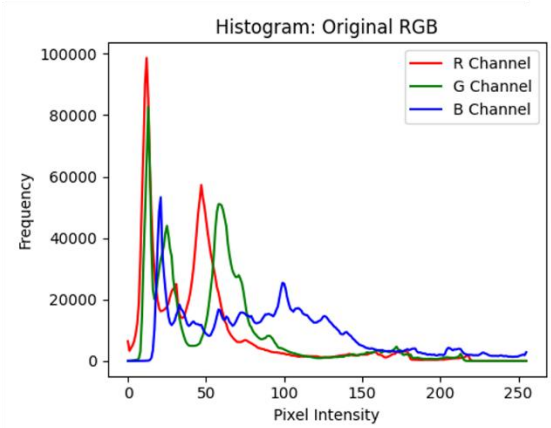
Python Code:

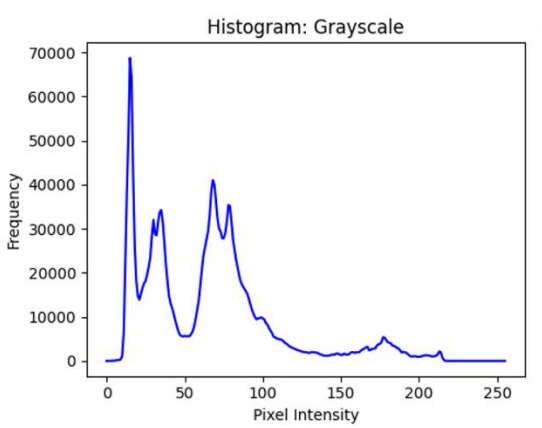
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os

def show_histogram(image, title):
    plt.figure(figsize=(6, 4))
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])
    plt.plot(hist, color='blue')
    plt.title(f'Histogram: {title}')
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')
    plt.show()

def show_image(image, title):
    plt.figure(figsize=(6, 6))
    plt.imshow(image, cmap='gray')
    plt.title(title)
    plt.axis('off')
    plt.show()

def show_combined_histograms(before_image, after_image, title_before, title_after,
is_rgb_before=False):
    plt.figure(figsize=(10, 4))

    plt.subplot(121)
    if is_rgb_before:
        colors = ('r', 'g', 'b')
        for i, color in enumerate(colors):
            hist = cv2.calcHist([before_image], [i], None, [256], [0, 256])
            plt.plot(hist, color=color, label=f'{color.upper()} Channel')
        plt.legend()
    else:
        hist_before = cv2.calcHist([before_image], [0], None, [256], [0, 256])
        plt.plot(hist_before, color='blue')
    plt.title(f'Histogram: {title_before}')
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')

    plt.subplot(122)
    hist_after = cv2.calcHist([after_image], [0], None, [256], [0, 256])
    plt.plot(hist_after, color='blue')
    plt.title(f'Histogram: {title_after}')
    plt.xlabel('Pixel Intensity')
    plt.ylabel('Frequency')
```

```
    plt.tight_layout()
    plt.show()

# Input Image path
image_path = 'Luffy.jpg'

if not os.path.exists(image_path):
    print(f"Error: Image file '{image_path}' not found")
    exit()

image = cv2.imread(image_path)
if image is None:
    print(f"Error: Could not load image at '{image_path}'")
    exit()

# i) Convert RGB Image to Grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
show_image(gray_image, 'Grayscale Image')
show_histogram(gray_image, 'Grayscale Image')
show_combined_histograms(image, gray_image, 'Original RGB', 'Grayscale',
is_rgb_before=True)

# ii) Change resolution to 512x512
resized_image = cv2.resize(gray_image, (512, 512), interpolation=cv2.INTER_AREA)
show_image(resized_image, 'Resized Image (512x512)')
show_histogram(resized_image, 'Resized Image')

# iii) Increase contrast using histogram equalization
equalized_image = cv2.equalizeHist(resized_image)
show_image(equalized_image, 'Equalized Image')
show_histogram(equalized_image, 'Equalized Image')

# iv) Display histograms before and after equalization together
show_combined_histograms(resized_image, equalized_image, 'Before Equalization',
'After Equalization', is_rgb_before=False)

# v) Apply Median filter
median_filtered = cv2.medianBlur(equalized_image, 5)
show_image(median_filtered, 'Median Filtered Image')
show_histogram(median_filtered, 'Median Filtered Image')

# Save all processed images
output_dir = 'd:/Desktop/Temp/Assignments/CV/'

cv2.imwrite(os.path.join(output_dir, 'grayscale_image.jpg'), gray_image)
cv2.imwrite(os.path.join(output_dir, 'resized_image.jpg'), resized_image)
cv2.imwrite(os.path.join(output_dir, 'equalized_image.jpg'), equalized_image)
cv2.imwrite(os.path.join(output_dir, 'median_filtered_image.jpg'), median_filtered)

print("All images processed and saved successfully.")
```

Output:

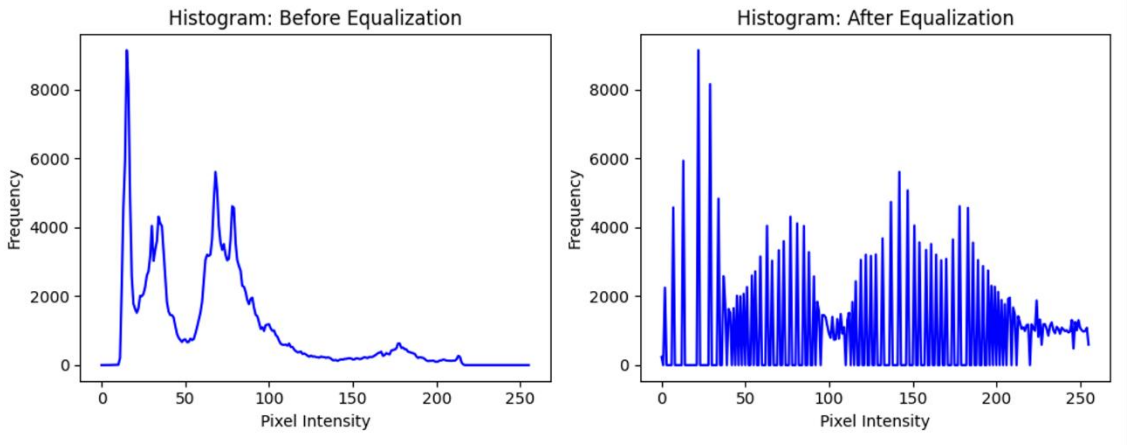

Grayscale Image

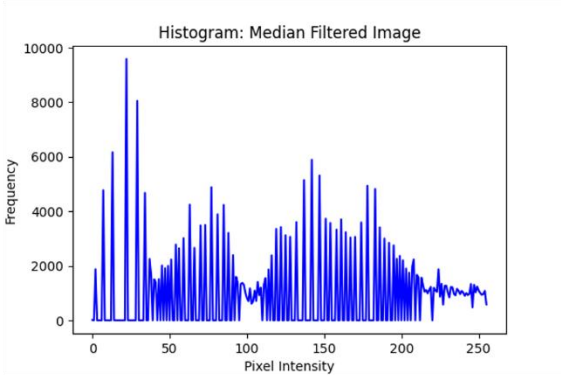

Histogram: Original RGB
R Channel
G Channel
B Channel
Frequency
Pixel Intensity
0
20000
40000
60000
80000
100000
0
50
100
150
200
250


Histogram: Grayscale
Frequency
Pixel Intensity
0
10000
20000
30000
40000
50000
60000
70000
0
50
100
150
200
250


Resized Image (512x512)


Equalized Image





1.2 Write a detailed note on Deep Learning for Computer Vision

Deep Learning for Computer Vision

Deep Learning (DL) is a subset of machine learning that leverages artificial neural networks, particularly deep neural networks with multiple layers, to model and solve complex tasks. In computer vision, DL has revolutionized the ability to process, analyze, and interpret visual data, enabling machines to perform tasks such as image classification, object detection, image segmentation, and facial recognition with unprecedented accuracy. This note provides an overview of deep learning in computer vision, covering its principles, architectures, applications, and key advancements.

Principles of Deep Learning in Computer Vision

Deep learning models for computer vision are primarily based on **Convolutional Neural Networks (CNNs)**, which are designed to process grid-like data, such as images. CNNs mimic the human visual system by learning hierarchical feature representations from raw pixel data. The key components of CNNs include:

- **Convolutional Layers**: Apply convolution operations to extract features like edges, textures, or patterns using learnable filters.
- **Pooling Layers**: Reduce spatial dimensions (e.g., max pooling) to decrease computational complexity and enhance translation invariance.

- **Activation Functions**: Introduce non-linearity (e.g., ReLU) to enable the network to learn complex patterns.
- **Fully Connected Layers**: Combine high-level features for tasks like classification.
- **Loss Functions**: Guide training by measuring the difference between predicted and actual outputs (e.g., cross-entropy for classification).

Other architectures, such as **Recurrent Neural Networks (RNNs)** for video analysis, **Transformers** for image understanding, and **Generative Adversarial Networks (GANs)** for image synthesis, have also become integral to computer vision.

Key Architectures

Several landmark architectures have shaped deep learning for computer vision:

- **LeNet (1998)**: One of the earliest CNNs, used for handwritten digit recognition.
- **AlexNet (2012)**: Popularized CNNs by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with deep layers and techniques like dropout and ReLU.
- **VGG (2014)**: Introduced deeper networks with smaller filters, emphasizing depth in CNNs.
- **ResNet (2015)**: Introduced residual connections to address vanishing gradients, enabling very deep networks (e.g., 152 layers).
- **Inception (GoogleNet, 2014)**: Utilized inception modules to capture multi-scale features efficiently.
- **YOLO (You Only Look Once, 2016)**: Enabled real-time object detection by treating detection as a single regression problem.
- **Vision Transformers (ViT, 2020)**: Adapted transformers from NLP to vision tasks, treating images as sequences of patches.

Training and Optimization

Training deep learning models involves feeding large datasets through the network, adjusting weights using backpropagation, and optimizing a loss function with algorithms like Stochastic Gradient Descent (SGD) or Adam. Key considerations include:

- **Datasets**: Large, annotated datasets like ImageNet, COCO, and Open Images are critical for training robust models.
- **Data Augmentation**: Techniques like rotation, flipping, and color jittering improve generalization.
- **Transfer Learning**: Pre-trained models (e.g., on ImageNet) are fine-tuned for specific tasks to reduce training time and data requirements.
- **Regularization**: Methods like dropout, weight decay, and batch normalization prevent overfitting.

Applications

Deep learning has enabled breakthroughs in numerous computer vision applications:

- **Image Classification**: Assigning labels to images (e.g., identifying whether an image contains a cat or dog).
- **Object Detection**: Identifying and localizing objects within an image (e.g., detecting cars in traffic camera footage).
- **Image Segmentation**: Partitioning an image into meaningful regions (e.g., semantic segmentation for medical imaging).
- **Facial Recognition**: Identifying or verifying individuals in images or videos.
- **Image Generation**: Creating realistic images using GANs or diffusion models (e.g., generating artwork).
- **Video Analysis**: Tracking objects, analyzing actions, or summarizing video content.

- **Autonomous Driving**: Enabling vehicles to detect lanes, pedestrians, and traffic signs.
- **Medical Imaging**: Assisting in diagnosing diseases from X-rays, MRIs, or CT scans.

Advancements and Trends

Recent advancements in deep learning for computer vision include:

- **Self-Supervised Learning**: Learning representations from unlabeled data (e.g., SimCLR, DINO), reducing reliance on annotated datasets.
- **Efficient Architectures**: Models like MobileNet and EfficientNet optimize for performance on resource-constrained devices.
- **Transformers in Vision**: Vision Transformers and variants (e.g., Swin Transformer) outperform CNNs in some tasks by leveraging attention mechanisms.
- **Diffusion Models**: Generating high-quality images by iteratively denoising data, rivaling GANs.
- **Explainable AI**: Techniques to interpret model decisions, crucial for trust in applications like medical diagnosis.

Conclusion

Deep learning has transformed computer vision by enabling machines to achieve human-like performance in visual tasks. Its success is driven by advances in neural architectures, large-scale datasets, and computational power. As research progresses, deep learning continues to push the boundaries of what is possible in computer vision, with applications spanning industries from healthcare to autonomous systems.

1.3 Discuss the challenges in implementing Deep Learning for Computer Vision to solve the real word problems.

Challenges in Implementing Deep Learning for Computer Vision to Solve Real-World Problems

While deep learning has revolutionized computer vision, implementing it to solve real-world problems presents several challenges. These challenges stem from technical, practical, and ethical considerations, impacting the deployment of robust, scalable, and equitable solutions. Below is a detailed discussion of these challenges.

1. Data-Related Challenges

a. Data Quantity and Quality

- **Issue**: Deep learning models require large amounts of labelled data to achieve high performance. For example, training an object detection model for autonomous driving needs millions of annotated images.
- **Impact**: In domains like medical imaging, annotated data is scarce due to privacy concerns and the need for expert labelling. Poor-quality data (e.g., noisy, incomplete, or biased) leads to suboptimal models.
- **Prevention**: Use data augmentation, transfer learning, or self-supervised learning to reduce data requirements. Synthetic data generation (e.g., using GANs) can also help.

b. Data Bias and Representation

- **Issue**: Datasets often reflect biases in their collection process (e.g., underrepresentation of certain demographics in facial recognition datasets).
- **Impact**: Biased models can produce unfair or inaccurate results, such as misidentifying individuals from underrepresented groups or failing to detect objects in diverse environments.

- **Prevention**: Curate diverse datasets, apply fairness-aware algorithms, and use techniques like adversarial training to reduce bias.

c. Domain Shift

- **Issue**: Models trained on one dataset may perform poorly when applied to different environments (e.g., a model trained on sunny-day images failing in foggy conditions).
- **Impact**: Real-world applications like autonomous driving or surveillance require robustness to varying conditions (lighting, weather, camera angles).
- **Prevention**: Domain adaptation, robust data augmentation, and continual learning can help models generalize across domains.

2. Computational and Resource Constraints

a. High Computational Requirements

- **Issue**: Training deep learning models, especially large CNNs or transformers, requires significant computational resources (e.g., GPUs, TPUs).
- **Impact**: High costs limit accessibility for smaller organizations or resource-constrained settings, such as deploying models on edge devices (e.g., mobile phones, IoT devices).
- **Prevention**: Use efficient architectures (e.g., MobileNet, EfficientNet), model compression (e.g., pruning, quantization), or cloud-based training solutions.

b. Real-Time Processing

- **Issue**: Many real-world applications, such as autonomous driving or video surveillance, require real-time inference.
- **Impact**: Complex models with high latency are impractical for time-sensitive tasks.
- **Prevention**: Optimize models for speed (e.g., YOLO for real-time object detection) and deploy on specialized hardware like edge TPUs.

3. Model Robustness and Generalization

a. Overfitting and Generalization

- **Issue**: Deep learning models can overfit to training data, failing to generalize to new, unseen data.
- **Impact**: Overfitted models perform poorly in real-world scenarios with variability (e.g., different camera resolutions or lighting conditions).
- **Prevention**: Use regularization techniques (e.g., dropout, weight decay), data augmentation, and diverse training datasets.

b. Adversarial Attacks

- **Issue**: Deep learning models are vulnerable to adversarial examples—inputs with imperceptible perturbations that cause misclassification.
- **Impact**: In security-critical applications (e.g., facial recognition, autonomous vehicles), adversarial attacks can lead to catastrophic failures.
- **Prevention**: Implement adversarial training, robust optimization, or input preprocessing to enhance model robustness.

4. Interpretability and Explainability

- **Issue**: Deep learning models are often "black boxes," making it difficult to understand their decision-making process.
- **Impact**: In applications like medical diagnosis or legal systems, lack of interpretability reduces trust and adoption.
- **Prevention**: Use explainable AI techniques (e.g., Grad-CAM, SHAP) to visualize model decisions and develop interpretable System table models.

5. Ethical and Societal Concerns

a. Privacy and Surveillance

- **Issue**: Computer vision applications like facial recognition or crowd monitoring raise privacy concerns.
- **Impact**: Widespread use can lead to misuse, such as unauthorized tracking or profiling.
- **Prevention**: Implement strict data privacy policies, anonymize data, and adhere to regulations like GDPR.

b. Ethical Implications of Bias

- **Issue**: Biased models can lead to unfair outcomes, such as discriminatory hiring or law enforcement practices.
- **Impact**: Reinforces societal inequalities and erodes public trust.
- **Prevention**: Regular audits, transparent model development, and inclusive dataset creation.

c. Misuse of Technology

- **Issue**: Deep learning can be used for malicious purposes, such as deepfake generation or autonomous weapons.
- **Impact**: Poses risks to security, trust, and safety.
- **Prevention**: Develop ethical guidelines, regulatory frameworks, and detection tools for malicious applications.

6. Scalability and Deployment Challenges

a. Integration into Real-World Systems

- **Issue**: Deploying CV models into existing infrastructure (e.g., hospital systems, autonomous vehicles) requires compatibility and reliability.
- **Impact**: Incompatibilities or errors can disrupt operations or lead to safety issues.
- **Prevention**: Use modular architectures, standardized APIs, and rigorous testing.

b. Maintenance and Updating

- **Issue**: Models need regular updates to remain effective as data distributions change (e.g., new types of objects in detection systems).
- **Impact**: Outdated models can degrade performance or become obsolete.
- **Prevention**: Implement continual learning, automated retraining pipelines, and version control.

Conclusion

Implementing deep learning for computer vision in real-world applications is challenging due to data limitations, computational demands, robustness issues, interpretability needs, ethical concerns, and deployment complexities. Addressing these challenges requires interdisciplinary efforts, including advances in algorithms, hardware, data curation, and ethical frameworks to ensure reliable, fair, and impactful solutions.

Benefits Derived from Solving the Assignment

Completing this assignment provided several valuable benefits, enhancing both technical and analytical skills in computer vision and deep learning:

- **Practical Experience with OpenCV**: Developing the Python script to perform image processing tasks (RGB to grayscale conversion, resizing, histogram equalization, histogram display, and median filtering) deepened my understanding of OpenCV's functionality. Troubleshooting the cv2.imshow error and adapting the code to use Matplotlib for display improved my ability to handle real-world implementation challenges, such as GUI backend issues, and reinforced the importance of robust error handling and alternative solutions.

- **Strengthened Programming and Debugging Skills**: Writing and refining the code to meet specific requirements (e.g., separate displays for each step, combined histograms) honed my Python programming skills. Addressing user feedback, such as ensuring the correct image path and sequential displays, enhanced my debugging and iterative development capabilities, making me more adept at delivering user-aligned solutions.
- **In-Depth Knowledge of Deep Learning for Computer Vision**: Researching and articulating a detailed note on deep learning for computer vision expanded my theoretical understanding of key concepts, including CNN architectures (e.g., AlexNet, ResNet), training techniques (e.g., transfer learning), and emerging trends (e.g., Vision Transformers, diffusion models). This exercise solidified my grasp of how deep learning drives advancements in visual tasks like object detection and image segmentation.
- **Critical Analysis of Real-World Challenges**: Exploring the challenges in implementing deep learning for computer vision, such as data biases, computational constraints, and ethical concerns, sharpened my ability to think critically about the practical limitations of AI technologies. This analysis highlighted the importance of interdisciplinary approaches, combining technical solutions (e.g., domain adaptation) with ethical considerations (e.g., fairness audits).
- **Enhanced Communication and Documentation Skills**: Structuring the responses in a clear, organized manner, with markdown artifacts, improved my ability to communicate complex technical concepts effectively. Providing detailed explanations, verifying requirements, and offering actionable instructions (e.g., installation steps, troubleshooting tips) reinforced the value of clear documentation in technical projects.

Overall, this assignment was a comprehensive learning experience, blending hands-on coding, theoretical exploration, and critical thinking. It equipped me with practical skills in image processing, a deeper understanding of deep learning, and the ability to address real-world complexities, preparing me for future challenges in computer vision and AI development.

References

- ✓ Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
- ✓ He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- ✓ Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*.
- ✓ LeCun, Y., Boser, B., Denker, J. S., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation, 1*(4), 541-551.
- ✓ Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. *International Conference on Machine Learning (ICML)*.
- ✓ Buolamwini, J., & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. *Conference on Fairness, Accountability, and Transparency (FAT)*.
- ✓ Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*.