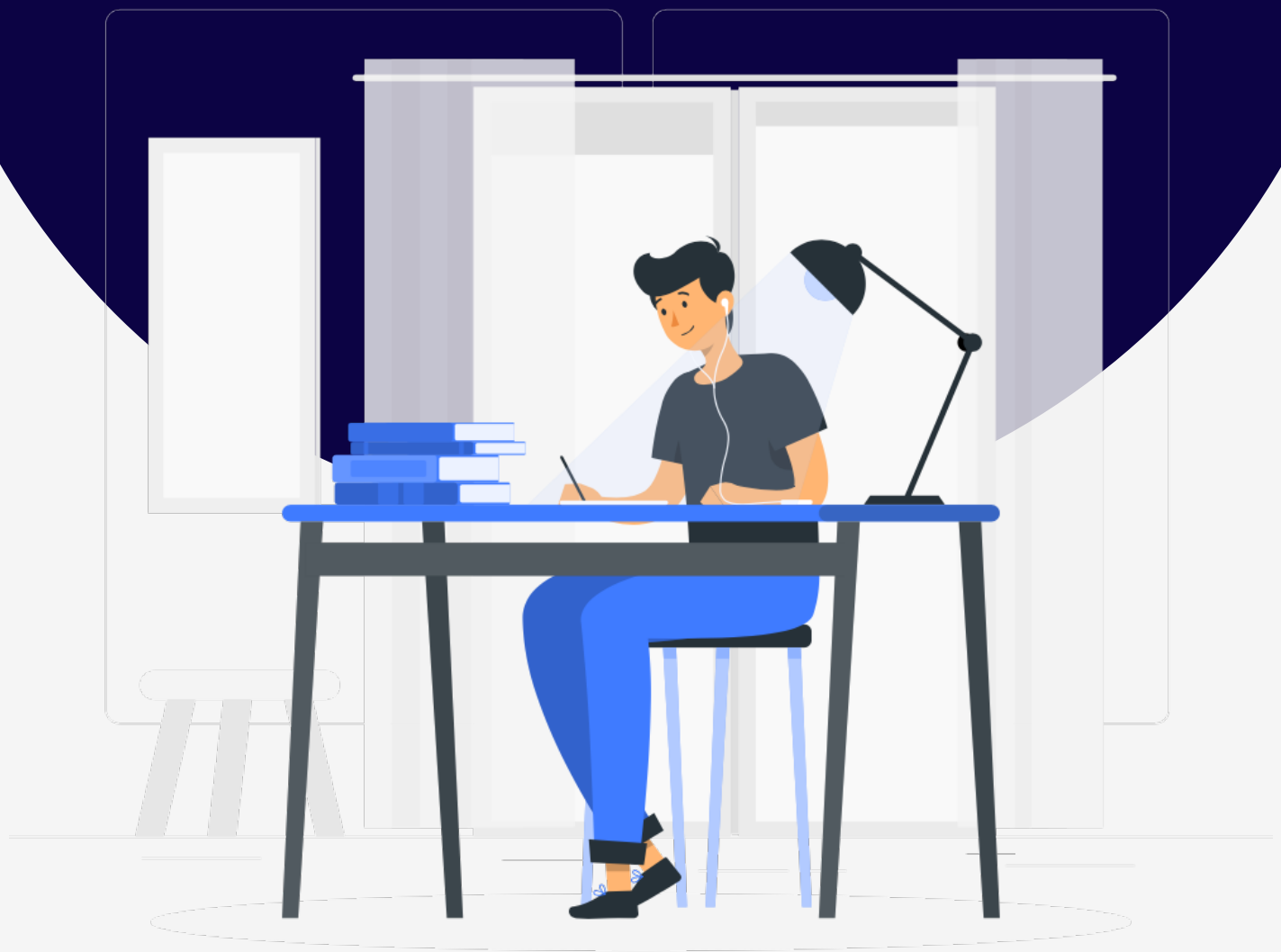
RCNN Architecture Assignment



1. **What are the objectives of using Selective Search in R-CNN?**
2. **Explain the following phases involved in R-CNN:**
   a. Region proposal
   b. Warping and Resizing
   c. Pre trained CNN architecture
   d. Pre Trained SVM models
   e. Clean up
   f. Implementation of bounding box
3. **What are the possible pre trained CNNs we can use in Pre trained CNN architecture?**
4. **How is SVM implemented in the R-CNN framework?**
5. **How does Non-maximum Suppression work?**
6. **How Fast R-CNN is better than R-CNN?**
7. **Using mathematical intuition, explain ROI pooling in Fast R-CNN.**
8. **Explain the following processes:**
   a. ROI Projection
   b. ROI pooling
9. **In comparison with R-CNN, why did the object classifier activation function change in Fast R-CNN?**
10. **What major changes in Faster R-CNN compared to Fast R-CNN?**
11. **Explain the concept of Anchor box.**
12. **Implement Faster R-CNN using 2017 COCO dataset (link: https://cocodataset.org/#download) i.e. Train dataset, Val dataset and Test dataset. You can use a pre-trained backbone network like ResNet or VGG for feature extraction. For reference implement the following steps:**

    a. Dataset Preparation:

    i. **Download and preprocess the COCO dataset, including the annotations and images.**
    ii. **Split the dataset into training and validation sets.**

    b. Model Architecture:

    i. Build a Faster R-CNN model architecture using a pre-trained backbone (e.g., ResNet-50) for feature extraction.
    ii. Customise the RPN (Region Proposal Network) and RCNN (Region-based Convolutional Neural Network) heads as necessary.

    c. Training:

    i. Train the Faster R-CNN model on the training dataset.
    ii. Implement a loss function that combines classification and regression losses.
    iii. Utilise data augmentation techniques such as random cropping, flipping, and scaling to improve model robustness.

    d. Validation:

    i. Evaluate the trained model on the validation dataset.
    ii. Calculate and report evaluation metrics such as mAP (mean Average Precision) for object detection.

    e. Inference:

    i. Implement an inference pipeline to perform object detection on new images.
    ii. Visualise the detected objects and their bounding boxes on test images.

    f. Optional Enhancements:

    i. Implement techniques like non-maximum suppression (NMS) to filter duplicate detections.
    ii. Fine-tune the model or experiment with different backbone networks to improve performance.

Submission Guidelines:

- Answer all the questions in a single Jupyter Notebook file (.ipynb).
- Include necessary code, comments, and explanations to support your answers and implementation.
- Ensure the notebook runs without errors and is well-organized.
- Create a GitHub repository to host your assignment files.
- Rename the Jupyter Notebook file using the format "date_month_topic.ipynb" (e.g., "21st_September_RCNN Architecture.ipynb").
- Place the Jupyter Notebook file in the repository.
- Commit and push any additional files or resources required to run your code (if applicable) to the repository.
- Ensure the repository is publicly accessible.
- Submit the link to your GitHub repository as the assignment submission.

Grading Criteria:

- Understanding and completeness of answers: 40%
- Clarity and depth of explanations: 25%
- Correct implementation and evaluation of optimizer techniques: 15%
- Analysis and comparison of different optimizers: 10%
- Proper code implementation and organization: 10%

Note: Create your assignment in Jupyter notebook and upload it to GitHub & share that uploaded assignment file link through your dashboard. Make sure the repository is public.