IIIT-Bangalore

Visual Recognition

AIM - 825

Professors:
Prof. Viswanath G
Prof. Sushree Behera

Submitted by:
Madhav Girdhar (IMT2022009)
Siddhesh Deshpande (IMT2022080)
Krish Patel (IMT2022097)

Task-1 : Binary Classification Using Handcrafted Features and ML Classifiers

Introduction

The objective of this project is to develop a computer vision-based system to detect face masks in images. The primary goal is to classify images into two categories: masked and unmasked faces, using machine learning techniques. The study evaluates the effectiveness of handcrafted feature extraction methods, comparing the performance of Support Vector Machine (SVM), Random Forest, and Neural Networks in classification accuracy.

Dataset

1. The data set contains two categories:
   (a) With Mask: Images of people wearing face masks.
   (b) Without Mask: Images of people without face masks.
2. The images are organized into separate folders for easy access and processing.
3. The source of the data set is mentioned at the end of the report.

Methodology

1. Feature Extraction Using HOG (Histogram of Oriented Gradients):
   - Convert images to grayscale for uniform feature extraction.
   - Apply HOG to extract texture and edge-based features from the images.
2. Load Dataset into X and Y :
   - Append extract feature in X and a value of 1 for masked image and 0 for non-masked image in Y.
3. Standardize the data using a Standard Scaler and split it into training(80%) and testing(20%) sets to evaluate the model's accuracy.
4. Apply different models such as SVM, Random Forest, and Neural Network, and compare their performance and accuracy.

Results

1. Support Vector Machine:
    - Accuracy : 0.9426
    - Best Parameter:
        (a) Kernel = rbf
        (b) C = 10
        (c) gamma = scale
2. Random Forest:
    - Accuracy : 0.9133
    - Best Parameter:
        (a) max_depth = 20
        (b) min_samples_leaf = 2
        (c) min_samples_split = 5
        (d) n_estimators = 200
3. Neural Network:
    - Accuracy: 0.9353
    - Network Architecture:
        (a) First hidden layer : 64-neurons and Activation function : relu
        (b) Second hidden layer : 32-neurons and Activation function : relu
        (c) Output layer : 2-neurons and Activation function : softmax
        (d) Optimizer : adam

Observation and Analysis

- SVM performed well with HOG features but slightly underperformed compared to CNN-based approaches, achieving an accuracy of 94.26%. The RBF kernel with C=10 and gamma=scale provided optimal results.
- Random Forest had lower accuracy (91.33%) due to its inability to capture spatial features as effectively as deep learning models. However, tuning hyperparameters such as max_depth=20 and n_estimators=200 improved performance.
- Neural Networks outperformed other models with an accuracy of 93.53%, benefiting from deeper feature extraction. The two-layer architecture with ReLU activations and Adam optimizer contributed to this success.
- Challenges faced:

  - The handcrafted features used in SVM and Random Forest models had limitations in capturing complex patterns, while Neural Networks required careful tuning to avoid overfitting.
  - Some images lacked sharpness due to low resolution, making it difficult for the feature extraction methods (HOG and SIFT) to capture meaningful patterns.
  - Training an SVM with large datasets was time-consuming, especially when using the RBF kernel, as it scales poorly with increasing data points.

- Challenges addressed:
  - Handcrafted Feature Limitations: To improve feature representation, we experimented with deep learning-based feature extraction using CNNs which is done in task-2.
  - Low-Resolution Images: We applied image pre-processing techniques such as resizing, histogram equalization to enhance image quality before feature extraction.

- Traditional machine learning classifiers like SVM and Random Forest rely on handcrafted feature extraction methods such as HOG and SIFT to distinguish between masked and unmasked faces. These approaches require manual feature selection and may struggle with complex patterns. In contrast, CNN-based techniques automatically learn hierarchical features directly from raw image data, making them more effective for capturing spatial and texture details. CNNs typically achieve higher accuracy but require larger datasets and more computational power for training which is more explained in next part of report.

Task-2 : Binary Classification Using CNN.

Introduction

The objective of this project is to develop a computer vision-based system for face mask detection in images. The primary goal is to classify images using Convolutional Neural Networks (CNNs), which automatically learn features without the need for handcrafted feature extraction.

Dataset

The dataset used is the same as mentioned in Task 1 of the report.

Methodology

1. Transform the Data using the transforms module of torchvision.The Following transformations were applied

   - The image was resized to 224x224.
   - RandomHorizontalFlip was performed to flip few images randomly.
   - Similarly RandomRotation was applied to rotate few of the images.
   - The Image Data was converted in tensor format.
   - The pixel values were normalized with mean=0.5 and std=0.5 as the parameters for the transforms.Normalize() Function.
   - ColorJitter was used to randomly change the brightness,contrast.

   All these steps help model to generalize well .

2. The Data was split into Training ,Validation and Test Set . 80% of the data was used for training and 10% for validation and remaining 10$ for testing.

3. The `ImageClassificationBase` class is a helper class for training and evaluating image classification models in PyTorch. It provides essential methods for handling training and validation:

   - `training_step(batch)`: Computes the loss for a given batch using cross-entropy loss.
   - `validation_step(batch)`: Computes both loss and accuracy for a batch during validation.
   - `validation_epoch_end(outputs)`: Aggregates loss and accuracy across all validation batches in an epoch.
   - `epoch_end(epoch, result)`: Prints the validation loss and accuracy at the end of each epoch.

4. The Following is the Architecture of the CNN that was used for the classification Task.
    - There are 3 Convolutional Blocks in the CNN model and 3 Fully Connected Layers.
    - Each block consist of 2 Convolutional Layers and a Max Pooling The Kernel of 3x3 dimension was used across all the convolutional layers with a stride of 1 and padding 1.Maxpool was done with windows of size 2x2.Layer is present after these two layers. The layers in the block are stacked in following manner.
        - Convolutional Layer
        - Activation Function
        - Convolutional layer
        - Activation Function.
        - MaxPool Layer.
    - The number of channels were increased in the order 3,32,64,128,128,256,256 as we go deep in the network.
    - Finally the output of these convolutional blocks was flattened was conenected to FC layers.
    - First FC layer consisted of 1024 neurons, second had 512 and third and final layer had 2 output neurons as this is a binary classification task.Activation Function was used between two consecutive layers except after the last layer.
5. For Training the Model two functions evaluate and fit were defined also the data was loaded into the device used(cpu/gpu) using the DeviceDataLoader.
    - fit(num_epochs, lr, model, train_loader, val_loader, opt_func)
    - evaluate(model, val_loader)
6. Some parameters like no_of_epochs,Optimizer and learning rate were intialized as variables as they need to be specified for training .
7. The Model was trained using the fit function that was created .
8. After Training model was evaluated on the test data using evaluate function.

Hyperparameters and Experiments

We have experimented with a variety of hyperparameters like learning rate,optimizer,batch size and activation function.Following are the experiments and their results.

Experiment 1

- Learning Rate:0.0001
- Batch Size : 32
- Activation Function: Relu
- number of epochs:10
- Optmizer:Adam
- Accuracy on test data : 97.99%

Experiment 2

- Learning Rate:0.01
- Batch Size : 32
- Activation Function: relu
- number of epochs:10
- Optmizer:SGD
- Accuracy on test data : 86.80%

Experiment 3

- Learning Rate:0.001
- Batch Size : 64
- Activation Function: relu
- number of epochs:10
- Optmizer:Adam
- Accuracy on test data : 95.89%

Experiment 4

- Learning Rate:0.001
- Batch Size : 32
- Activation Function: relu
- number of epochs:10
- Optmizer:AdamW
- Accuracy on test data : 95.53%

Experiment 5

- Learning Rate:0.001
- Batch Size : 64
- Activation Function: Sigmoid
- number of epochs:10
- Optmizer:Adam
- Accuracy on test data : 47.76%

Observation and Analysis

- The Relu Activation Function with Adam Optimizer and batch size of 32 and learning rate 0.0001 gave the highest accuracy.This shows that adam optimizer works well with low learning rate.
- With other parameters being similar we observed that sigmoid activation Function performed very poorly as compared to relu.
- Also when SGD Optimzer was used there was a significant decrease in accuracy of the model as it performed extrememly poor with this optimizer when compared to Adam and AdamW.
- Many Other Activation functions like Gelu , LeakyRelu gave accuracy around 94-95% with the same parameters as when we got the max accuracy with relu.

Task-3 : Region Segmentation Using Traditional Techniques

Introduction

Image segmentation plays a critical role in computer vision tasks, especially in scenarios where identifying the boundaries or specific regions of objects is required. In this project, before implementing deep learning models like U-Net, we explored classical image segmentation methods. These traditional approaches help in understanding how image features can be used for segmentation without the need for model training. The methods employed include Otsu's thresholding and segmentation using edge detection techniques.

Dataset

The dataset used for this exploration consists of input images that are read from a defined path. These images are of various objects where segmentation masks are intended to be created. The dataset contains images, with mask and without mask, their ground truth segmented data. We now try to deploy different traditional image segmentation techniques to compare the output with ground truth values.

Methodology

Otsu's Thresholding:

- The grayscale version of each input image is computed.
- Otsu's thresholding method is applied, which calculates an optimal threshold value by maximizing inter-class variance.
- A binary mask is generated from this thresholded image.
- Morphological closing operations are performed to fill small holes and remove noise.
- The binary mask is converted back to a 3-channel mask to align with the original image dimensions for visualization.

Segmentation using Edge Detection:

- Edge detection is performed using the Canny Edge Detector.
- The detected edges are then dilated to make them more prominent.
- The dilated edge maps are filled using contour filling methods to create binary segmentation masks.

- Morphological operations like closing and dilation are used to refine the mask and ensure continuity in detected regions.
- The binary masks are converted to 3-channel representations for better visualization alongside the original images.

Results

IoU value: 0.528

- Otsu's thresholding was successful in segmenting regions with clear intensity differences between objects and backgrounds.
- The resulting masks, after morphological operations, displayed good separation of objects.
- Edge detection-based segmentation was effective in highlighting object boundaries but required contour filling and morphological refinements to create complete object masks.
- Dilation helped in bridging gaps in detected edges and creating continuous boundaries.

Figure 1: Images and corresponding ground truth values

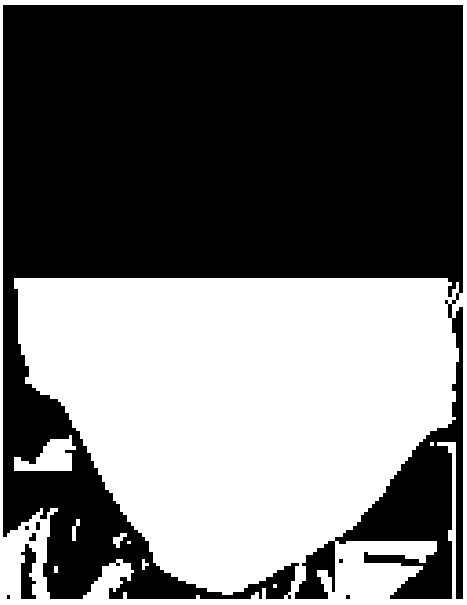

Figure 2: Segmented Mask Example

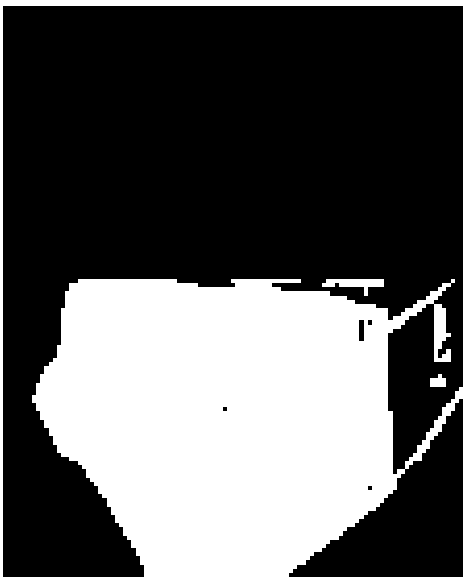

Figure 3: Segmented Mask Example

Observation and Analysis

- Otsu's thresholding works well for images with distinct grayscale intensity differences. However, it struggles with images where the background and object intensities are close.
- The use of morphological operations is crucial for refining the binary masks, reducing noise, and filling small holes.
- Edge detection techniques, particularly using Canny detectors, are very sensitive to parameter tuning (like threshold values).
- Contour filling and morphological operations are necessary to convert edge maps into usable masks.
- Classical methods can provide quick segmentation results without model training but lack robustness compared to deep learning approaches for complex images.

These observations establish a baseline understanding of traditional segmentation techniques before moving to learning-based methods like U-Net, providing context and comparison for the performance improvements achieved by deep models.

Task-4 : Mask Segmentation Using U-Net.

Introduction

Image segmentation is fundamental to many computer vision applications, particularly in medical imaging, autonomous driving, and satellite imagery analysis. After exploring classical methods like Otsu's thresholding and edge-based segmentation, this part of the project focuses on a more advanced, learning-based approach using the U-Net architecture for semantic segmentation.

Dataset

The dataset used is the same as mentioned in Task 3 of the report.

Methodology

1. Model Architecture (U-Net):
    - A custom U-Net architecture was implemented using TensorFlow and Keras.
    - The model consists of an encoder-decoder structure with skip connections.
    - Hyperparameters include the number of layers (depth), base number of filters, activation functions (ReLU, Leaky ReLU, or ELU), and optional dropout and batch normalization.
2. Model Compilation:
    - The model was compiled with the Adam optimizer and categorical cross-entropy loss function, suitable for multi-class segmentation.
3. Training Setup:
    - A custom training loop was defined to handle training and validation without external logging tools.
    - The model was trained for multiple epochs, with loss, accuracy, Intersection over Union (IoU), and Dice score metrics monitored.
4. Evaluation Metrics:
    - IoU was calculated per batch by comparing the intersection and union of predicted and true masks.
    - Dice score was used to evaluate overlap quality between predictions and ground truth.
5. Prediction Visualization:

- Post-training, a prediction grid was created to visualize sample images alongside their predicted and ground-truth segmentation masks.
- Denormalization was applied to the input images for better display.

Results

- The U-Net model demonstrated substantial improvement over classical methods in segmenting complex regions.
- Training loss consistently decreased, indicating the model's ability to learn pixel-level classifications.
- Validation IoU and Dice scores showed the model's strong generalization, with smooth and coherent boundaries in predictions.

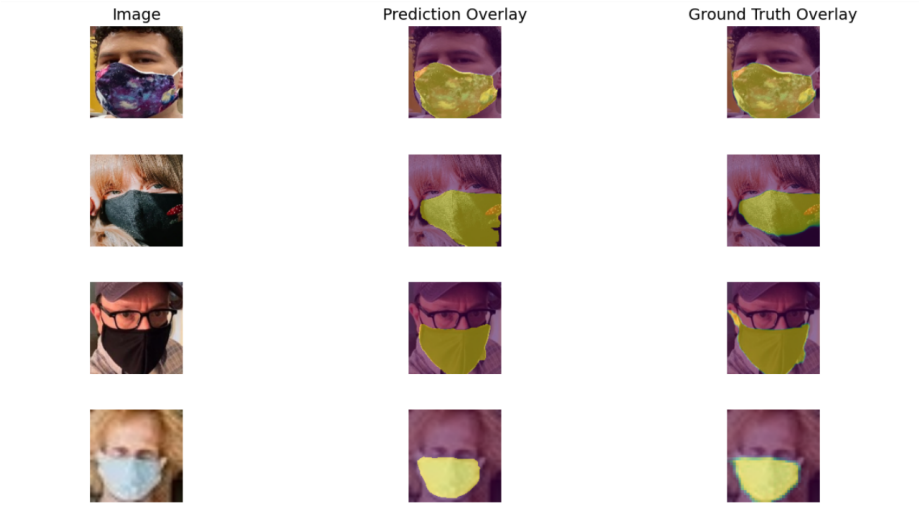

Figure 4: Images and corresponding ground truth values

Observation and Analysis

- Batch normalization and dropout helped mitigate overfitting and stabilized training.
- IoU and Dice scores confirmed that deep learning models outperform classical segmentation methods, especially in noisy or ambiguous scenarios.
- Visual overlays of predictions and ground truth highlighted the model's robustness in detecting fine structures and boundaries.

- The Following were the best accuracy,dice score and iou obtained.
  - Accuracy : 0.8446
  - IOU : 0.5892
  - Dice Score : 0.6472

Note

- Dataset : Click here
- Code : Click here
- Instructions on how to run the code are mentioned in the README file present in the repository.