Enhancing Image Classification in Low-Light Conditions Using Advanced CNN Techniques

Yuesong Huang | Wentao Jiang

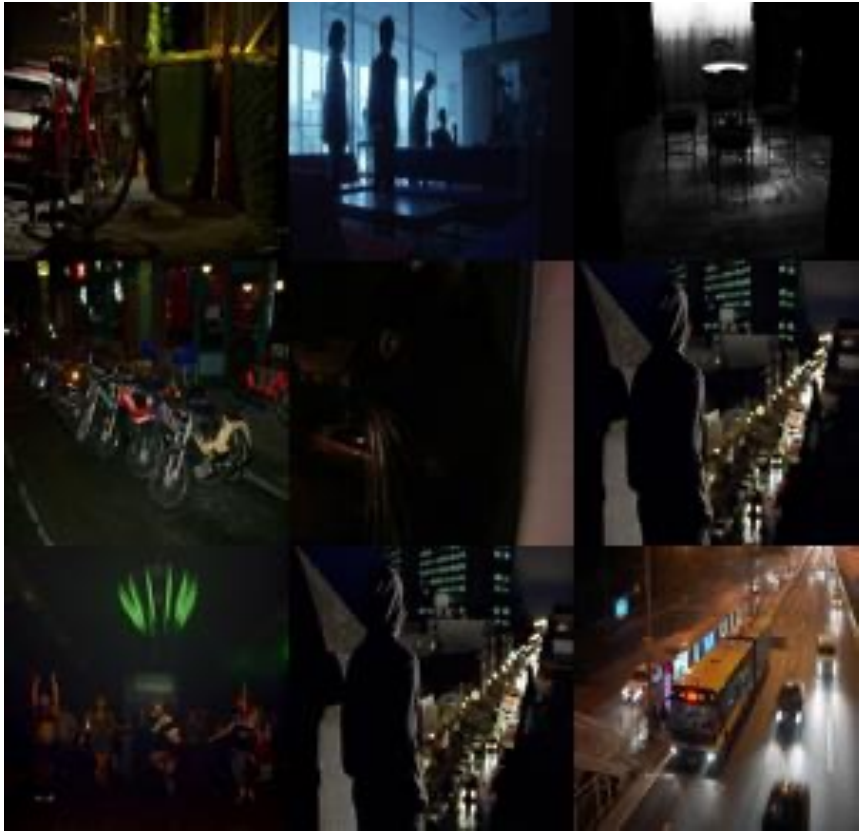
Nov 28, 2023

Introduction

- Challenge of image classification in low-light conditions
- Relevance to real-world applications
- Unique approach for multi-label classification

Dataset Overview

- **Dataset Source:**
  "*Exclusively Dark*" Image Dataset [1]
- **Classes/Labels:**
  - 7300 samples
  - 12 classes, i.e.,
  - Bicycle, Dog, People, etc.
- **Image Resolution:**
  - About 500 x 300
  - Transformed to 256 x 256



[1] Loh, Yuen Peng and Chan, Chee Seng (2019), *"Getting to Know Low-light Images with The Exclusively Dark Dataset"*, Computer Vision and Image Understanding

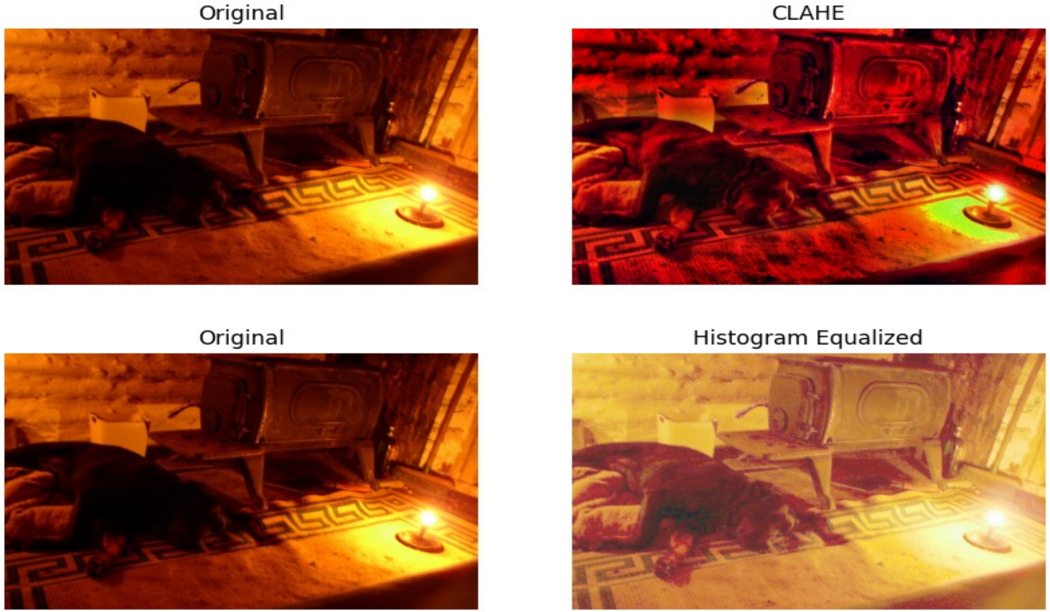
Initial Challenges

- Early focus on single-label classification
- Low *F1* scores due to unrecognized multi-label elements using **“micro”**
- *Example:* Images with multiple object incorrectly labeled and classified as single object
- Importance of precise image annotation highlighted for future improvements. (i.e., object detection)

```
# Histogram Equalization
class HETransform(object):
    def __call__(self, img_tensor):
        return self.histogram_equalization(img_tensor)

    def histogram_equalization(self, img_tensor):
        image = img_tensor.numpy()
        img_he = exposure.equalize_hist(image)
        he_tensor = torch.from_numpy(img_he)
        return he_tensor

# Contrast Limited Adaptive Histogram Equalization (CLAHE)
class CLAHETransform(object):
    def __call__(self, img_tensor):
        return self.clahe(img_tensor)

    def clahe(self, img_tensor):
        image = img_tensor.numpy()
        img_clahe = exposure.equalize_adapthist(image, clip_limit=
        clahe_tensor = torch.from_numpy(img_clahe)
        return clahe_tensor

# Dynamic Range Adjustment (DRA)
class DRATransform(object):
    def __call__(self, img_tensor):
        return self.dra(img_tensor)

    def dra(self, img_tensor):
        image = img_tensor.numpy()
        min_val = np.min(image)
        max_val = np.max(image)
        # Rescale to 0-1
        adjusted_image = (image - min_val) / (max_val - min_va
        adjusted_tensor = torch.from_numpy(adjusted_image)
        return adjusted_tensor

# Gaussian Blur
class NoiseCancellationTransform(object):
    def __call__(self, img_tensor):
        return self.noise_cancellation(img_tensor)

    def noise_cancellation(self, img_tensor):
        image = img_tensor.numpy()
        # Apply Gaussian filter for noise reduction
        # Adjust sigma as needed
        sigma = 1.0  # Standard deviation for Gaussian kerne
        filtered_image = filters.gaussian(image, sigma=sigma, mul
        filtered_tensor = torch.from_numpy(filtered_image)
        return filtered_tensor
```

Methodology and Improvements

- Feature engineering and image preprocessing steps



```
 First  use  CNN  as  base  case
ass  defaultCNN(nn.Module):
    def  __init__(self):
        super().__init__()
        self.conv1  =  nn.Conv2d(3,  64,  3,  1)
        self.pool1  =  nn.MaxPool2d(2,2)
        self.conv2  =  nn.Conv2d(64,  256,  5,  1)
        self.pool2  =  nn.MaxPool2d(2,2)
        self.fc  =  nn.Linear(256  *  61  *  61,  12)

    def  forward(self,  x):
        x  =  self.pool1(F.relu(self.conv1(x)))
        x  =  self.pool2(F.relu(self.conv2(x)))
        x  =  x.view(x.size(0),  -1)
        x  =  self.fc(x)
        return  x
```

```
vice  =  torch.device('cuda'  if  torch.cuda.is_available()  else  '

n_epochs  =  1
iterion  =  nn.BCEWithLogitsLoss()

f  initialize_model():
    model  =  defaultCNN()
    model  =  model.to(device)

    optimizer  =  optim.AdamW(model.parameters(),  lr=0.001)

    #  Added  a  new  schedular:  TODO:  reserved  for  early  sto
    scheduler  =  optim.lr_scheduler.StepLR(optimizer,  step_size=
    return  model,  optimizer,  scheduler
```

```
f  train_epoch(model,  optimizer,  criterion,  train_loader,  device,
    model.train()
    train_loss  =  0.0
    total  =  0
    for  i,  (inputs,  _,  labels)  in  enumerate(train_loader):
        inputs,  labels  =  inputs.to(device),  labels.to(devic
```

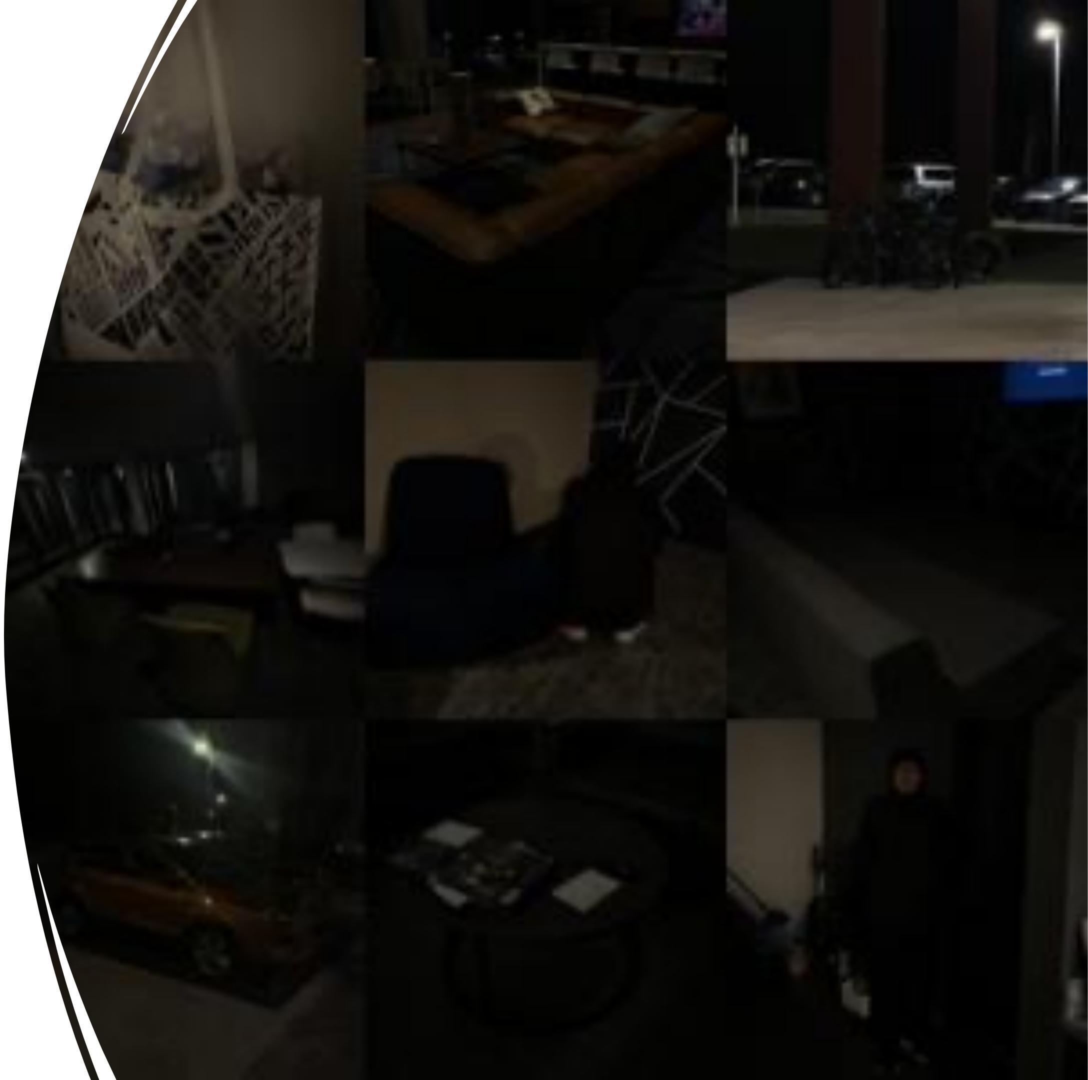
Methodology and Improvements

- Transition to enhanced model with **BCEWithLogitsLoss()**
- Handling multi-label classification effectively

image	class	x	y	w	h	fpath
2015_00426.jpg	Bicycle	98	408	142	229	./ExDark/Bicycle
2015_00426.jpg	People	104	279	104	306	./ExDark/Bicycle
2015_00426.jpg	People	84	210	88	149	./ExDark/Bicycle
2015_00426.jpg	People	157	210	84	218	./ExDark/Bicycle
2015_00426.jpg	People	226	196	73	263	./ExDark/Bicycle
2015_00426.jpg	People	248	216	108	336	./ExDark/Bicycle
2015_00426.jpg	People	309	170	92	360	./ExDark/Bicycle
2015_00073.jpg	Bicycle	156	256	191	232	./ExDark/Bicycle
2015_00073.jpg	Bicycle	41	199	24	52	./ExDark/Bicycle
2015_00073.jpg	Bicycle	251	192	27	32	./ExDark/Bicycle

Significant Milestone

- **Achievements**: Improved accuracy from *10%* to around *60%*; innovative use of *ResNet* and *VGG* backbones; feature engineering and preprocessing enhancements.
- Make our own **ExDark** Dataset!



Conclusion & Future Plans

- **Project Summary**: Significant advancement in *CNN* for low-light, multi-label image classification.
- **Real-World Impact**: Applications in security, autonomous navigation, and more.
- **Overall Learning**: Transformative experience in computer vision, pushing boundaries in low-light image classification

- Introducing object recognition
- Exploring bounding box-based object detection
- Plans for further accuracy enhancements

Thank You!

Q & A