PSI3472 Concepção e Implementação de Sistemas Eletrônicos Inteligentes
Segundo Semestre de 2024 Exercício-programa
Data de entrega: ~~13/10/2024~~ 20/10/2024 (domingo) até 23:59 horas

O livro [*Dive into Deep Learning*], disponível gratuitamente online, traz um conjunto de imagens *banana-detection* que pode ser usado para testar alguns conceitos de detecção de objetos.

https://d2l.ai/chapter_computer-vision/object-detection-dataset.html

Este conjunto consiste de 1000 imagens de treino e 100 de teste, todas coloridas com resolução 256×256. Em cada imagem, foi inserida a imagem de uma banana em posições, tamanhos e rotações aleatórias. Dois arquivos *label.csv* (um para treino e outro para teste) trazem as coordenadas verdadeiras da bounding box da banana em cada imagem. Por exemplo, o início do *label.csv* de treino é:

```
img_name,label,xmin,ymin,xmax,ymax
0.png,0,104,20,143,58
1.png,0,68,175,118,223
2.png,0,163,173,218,239
3.png,0,48,157,84,201
```

Em cada linha, aparece o nome de arquivo da imagem, seguido pelo rótulo “0” que indica a classe de objeto (neste caso, é sempre zero pois só há um tipo de objeto nas imagens). Os 4 últimos números indicam as coordenadas do bounding box (xmin, ymin, xmax, ymax). A figura abaixo mostra alguns exemplos de treino, com bananas delimitadas por bounding boxes verdadeiras.

(a) Imagem de treino 0.png,0,104,20,143,58

(b) Imagem de treino 1.png,0,68,175,118,223

(c) Imagem de teste 0.png,0,183,63,241,112

(d) Imagem de teste 1.png,0,26,86,79,133

Figura 1: Duas imagens de treino (a, b) e duas de teste (c, d) com bounding boxes verdadeiras de bananas.

Para baixar este conjunto do Google Drive para computador local, primeiro instale/atualize *gdown*:

```
$ pip3 install --upgrade gdown
```

Nota: Não é necessário fazer isto no Google Colab, pois *gdown* está pré-instalada.

Depois, execute o código abaixo (tanto no Colab como no computador local), que copiará o arquivo banana-detection.zip do Google Drive para o seu diretório atual e o descompactará.

```
import os; import gdown
nomeArq="banana-detection.zip"
if not os.path.exists(nomeArq):
  os.system("gdown 1J7IEFUlq8_ORnKGR3mthNjEQ5ecXmZY_")
os.system("unzip -u "+nomeArq)
```

Nota: Este conjunto pode ser baixado diretamente do site do livro [DiveintoDeepLearning] usando a biblioteca *d2l* disponibilizada pelo livro. Porém, isso modifica as versões de vários módulos Python do ambiente de trabalho.

O objetivo deste exercício é escrever um programa Python/Keras/Tensorflow que localiza o bounding box da banana em cada imagem de teste. Note que, quando estudamos classificação de imagens, a entrada era uma imagem e a saída era a classe do objeto (um número). Aqui, a entrada é uma imagem e a saída consiste de 4 números (xmin, ymin, xmax, ymax).

Nota: Escrever um detector de objetos deep learning "do zero" (e não simplesmente usar um modelo pronto) é uma tarefa bastante difícil. Porém, como cada imagem deste conjunto contém uma única banana, o problema fica bem mais simples.

Evidentemente, você **não** pode usar um modelo pronto de **detecção** de objetos. Porém, pode usar todas as outras técnicas que aprendemos no curso. Inclusive, se quiser, pode usar um modelo de **classificação** de imagens pré-treinado em ImageNet (como VGG, ResNet, EfficientNet, etc.) e usá-lo como modelo-base.

Informe o erro MAE de teste obtido (média da diferença absoluta entre as 400 coordenadas verdadeiras e as 400 obtidas pelo seu detector de bananas nas 100 imagens de teste). Este erro é a distância média entre as coordenadas verdadeiras e as coordenadas inferidas pelo seu modelo. A nota do EP dependerá desse erro. Se o erro cometido for maior que 14 pixels, receberá nota *máxima* 5,0. Se o erro for maior que 9 pixels, receberá nota *máxima* 7,5. Vamos ver quem consegue obter o menor erro.

A figura 2 mostra a saída do meu programa (baseado em rede neural convolucional de um único passo) que cometeu erro médio (MAE) de 2,79 pixels. Não implementei todas as técnicas que poderiam ser usadas para minimizar o erro. Portanto, certamente é possível obter erro médio ainda menor. Coloque no relatório uma imagem semelhante à figura 2 mostrando as detecções das bananas pelo seu programa nas 12 primeiras imagens de teste.

Nota: Tensorflow 2.17 (versão atual) está cometendo erro consideravelmente maior que Tensorflow 2.15 (versão antiga): respectivamente 3,8 e 2,8 pixels. Se for executar no Google Colab, desinstale Tensorflow 2.17 e instale alguma versão antiga.

Nota: Após detectar os bounding boxes usando deep learning, se quiser, pode fazer "ajuste fino" efetuando um segundo processamento com deep learning ou técnicas clássicas de processamento de imagens. Neste caso, informe o erro obtido após cada um dos dois passos.

Nota: Pode ser que usando as técnicas clássicas de processamento de imagens, como template matching, seja possível resolver este problema com erro muitíssimo pequeno.

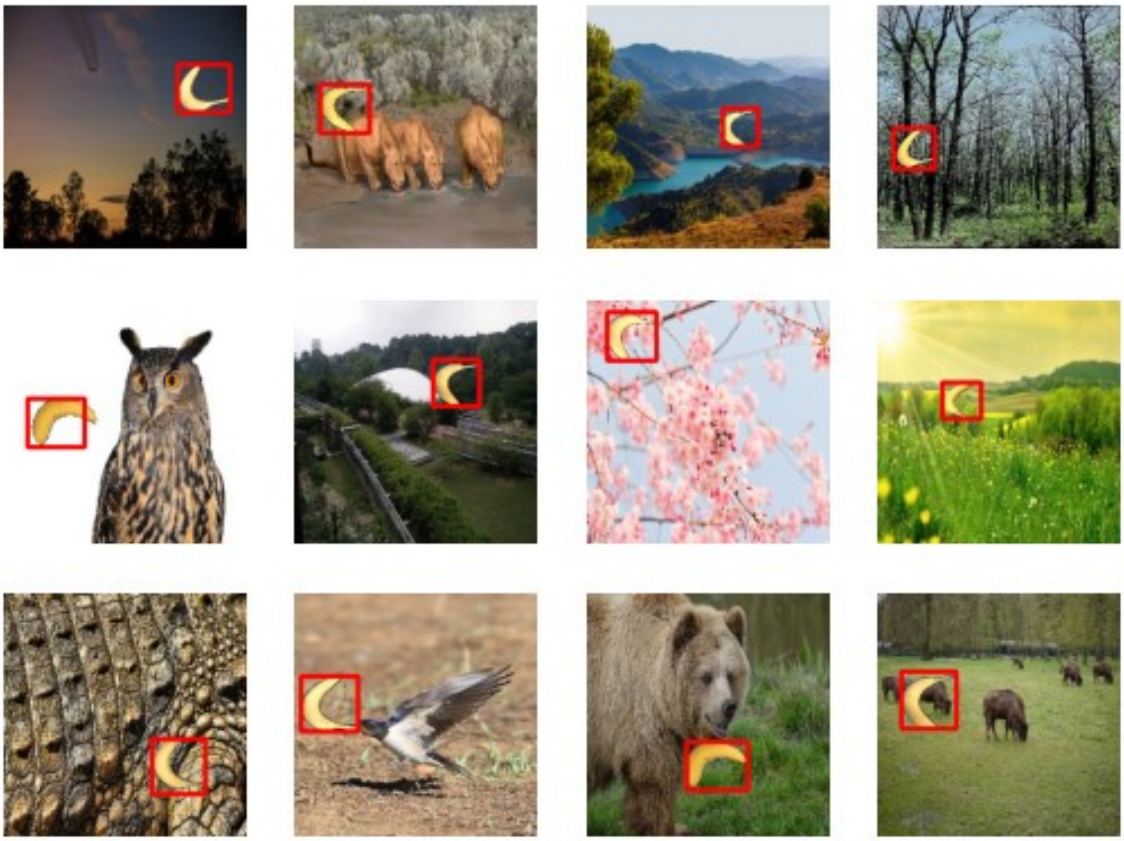

Figura 2: Saídas geradas pelo meu programa nas 12 primeiras imagens de teste. Erro médio foi 2,79 pixels.

Obs. 1: Este exercício pode ser resolvido em dupla ou individualmente.

Obs. 2: Em princípio, deve usar o mesmo ambiente de programação utilizado em aula (Python, Keras, OpenCV), computador local ou Google Colab. Se quiser utilizar outro ambiente, converse antes com o professor. Nota: Se quiser, pode entregar o programa em C++.

Obs. 3: Entregue os programas-fontes (banana.py ou banana.ipynb). Você pode enviar link para Colab ou Kaggle em vez dos programas: neste caso, assegure que todos os professores (Hae e Maurício - hae.kim@usp.br, mlisboa@usp.br) tenham acesso. Não entregue o dataset. Se você quiser explicar algo sobre alguma imagem, coloque-a no relatório e/ou mostre-a no vídeo.

Obs. 4: Entregue um documento PDF de no máximo 4 páginas (relatorio.pdf) descrevendo o funcionamento do seu programa, os resultados obtidos e as suas conclusões. O envio do relatório é obrigatório (veja o anexo). O relatório é um documento Word/LaTex/LibreOffice convertido para PDF. Um notebook Python anotado não será aceito como relatório.

Obs. 5: Entregue um vídeo de no máximo 90s explicando o funcionamento do seu programa, os resultados obtidos e as suas conclusões. Se EP foi feito em dupla, podem enviar dois vídeos (a duração total não deve superar 90s), cada membro explicando uma parte do EP. No vídeo deve aparecer em algum momento os rostos e os documentos dos membros do grupo. É necessário que o vídeo contenha áudio, pois é muito difícil entender um vídeo sem a explicação falada. O vídeo não pode ter sido acelerado para diminuir a duração, pois dificulta entender a fala. Vocês podem enviar o vídeo em si (.mkv, .mp4, .avi, etc.) ou um link para o vídeo (youtube, google drive, etc). No segundo caso, assegure que todos os professores (Hae e Maurício - hae.kim@usp.br, mlisboa@usp.br) tenham acesso.

Obs. 6: Não se esqueça de escrever/falar os nomes da dupla (ou do único aluno) em três lugares diferentes: no campo “comentários sobre o envio”, no início do vídeo e no início dos programas-fontes. Caso contrário, um dos alunos da dupla pode ficar sem a nota.

Obs. 7: Envie o material através de edisciplinas. Dentro do prazo, você pode substituir o material anterior por um novo.

Anexo: Relatórios dos exercícios programas

O mais importante numa comunicação escrita é que o leitor entenda, sem esforço e inequivocamente, o que o escritor quis dizer. O texto ficar "bonito" é um aspecto secundário. Se uma (pseudo) regra de escrita dificultar o entendimento do leitor, essa regra está indo contra a finalidade primária da comunicação. No site do governo americano [1], há regras denominadas de "plain language" para que comunicações governamentais sejam escritas de forma clara. As ideias por trás dessas regras podem ser usadas em outros domínios, como na escrita científica. Resumo abaixo algumas dessas ideias.

(1) Escreva para a sua audiência. No caso do relatório, a sua audiência será o professor ou o monitor que irá corrigir o seu exercício. Você deve enfocar na informação que o seu leitor quer conhecer. Não precisa escrever informações que são inúteis ou óbvias para o seu leitor.

(2) Organize a informação. Você é livre para organizar o relatório como achar melhor, porém sempre procurando facilitar o entendimento do leitor. Seja breve. Quebre o texto em seções com títulos claros. Use sentenças curtas. Elimine as frases e palavras que podem ser retiradas sem prejudicar o entendimento. Use sentenças em ordem direta (sujeito-verbo-predicado).

(3) Use o tempo verbal o mais simples possível. Evite "verbos ocultos" (por exemplo, substitua "precisamos realizar uma revisão das contas" para "precisamos rever as contas"; substitua "fiz o pagamento do seu salário" por "paguei o seu salário"). Evite cadeia longa de nomes, substituindo-os por verbos (em vez de "desenvolvimento de procedimento de proteção de segurança de trabalhadores de minas subterrâneas" escreva "desenvolvendo procedimentos para proteger a segurança dos trabalhadores em minas subterrâneas"). Minimize o uso de abreviações (para que o leitor não tenha que decorá-las). Use sempre o mesmo termo para se referir à mesma realidade, pois pode confundir o leitor se usar termos diferentes para se referir a uma mesma coisa. O relatório não é obra literária, não tem problema repetir várias vezes a mesma palavra.

(4) Use voz ativa. Deixe claro quem fez o quê. Se você utilizar oração com sujeito indeterminado ou na voz passiva, o leitor pode não entender quem foi o responsável (Ex: "Criou-se um novo algoritmo" - Quem criou? Você? Ou algum autor da literatura científica?). O site diz: "Passive voice obscures who is responsible for what and is one of the biggest problems with government writing."

(5) Use exemplos, diagramas, tabelas, figuras e listas. Ajudam bastante o entendimento.

O relatório deve conter pelo menos as seguintes informações:

Identificação: Seu nome, número USP, nome da disciplina, etc.

Ambiente de desenvolvimento utilizado: Em qual plataforma você desenvolveu o programa? Como o professor/monitor pode compilar o programa? Você utilizou que bibliotecas?

Operação: Como o professor/monitor pode executar o programa? Que argumentos são necessários para a execução do programa? Há parâmetros que devem ser configurados? Quais arquivos de entrada são necessários? Quais arquivos de saída são gerados?

Resultados Obtidos: Descreva os resultados obtidos. Qual é o tempo de processamento típico? O problema foi resolvido de forma satisfatória?

Referências: Referencie os materiais externos utilizados, como livros/artigos consultados, websites visitados, etc.

1 https://plainlanguage.gov/guidelines/