PSI-5796: Processamento e Análise de Imagens e Vídeos
Primeiro semestre de 2021 Exercício-programa Prof. Hae
Data de entrega: ~~13/06/2021 (domingo)~~ 16/06/2021 (quarta) até 24:00 horas

Obs. 1: Cada dia de atraso acarreta perda de 1 ponto no exercício.
Obs. 2: Este EP deve ser resolvido individualmente. EPs iguais receberão nota zero.
Obs. 3: Você não pode dar "copy-paste" de soluções completas encontradas na internet. Porém, evidentemente, pode usar trechos de programas encontrados na internet como exemplos.

O objetivo deste exercício é identificar se numa imagem aparece gato ou cachorro. Para isso, construa dois classificadores: (a) fazendo o treino "do zero" sem usar "transfer learning"; (b) usando "transfer learning".

No site https://www.kaggle.com/tongpython/cat-and-dog, há um banco de dado com 10.000 imagens, distribuídas em:

Treino: 4.000 imagens de gatos e 4.000 imagens de cachorros.
Teste: 1.000 imagens de gatos e 1.000 imagens de cachorros.

A figura abaixo mostra 10 primeiras imagens "em ordem alfabética" de gatos e cachorros para treino.

Figura 1: 10 primeiras imagens de treino de gatos e cachorros em ordem alfabética.

Nota: No BD, há alguns arquivos repetidos com nome da forma "*(1).jpg". Estes arquivos devem ser apagados para resultar em exatamente 10.000 imagens.

Nota: O aluno Rafael R. Lima deixou uma versão com as imagens duplicadas removidas em:
https://www.kaggle.com/rafaelrlima/cat-and-dog-clean-dataset

Faça quatro programas Python3/Keras (ou PyTorch): treino_zero.py, teste_zero.py, treino_transf.py e teste_transf.py.

1) O programa treino_zero.py deve fazer o treino “do zero” (isto é, sem usar transfer learning) utilizando as 4000+4000 imagens de treino. Se quiser, pode usar um subconjunto qualquer das imagens de treino para validação. Você deve gravar o arquivo zero.h5 (ou zero.pth) no diretório default com a rede treinada. Você pode usar “data augmentation” ou outras técnicas para diminuir a taxa de erro – só não pode usar “transfer learning”. O programa teste_zero.py deve ler zero.h5 e fazer predições, calculando a taxa de erro de teste e imprimindo as 10 primeiras imagens de gatos e 10 primeiras imagens de cachorros classificadas incorretamente.

Usando a rede “tipo LeNet”, sem fazer “data augmentation” e sem fazer muito esforço para diminuir erro, obtive taxa de erro de 19,95%. A figura abaixo mostra as 10 primeiras imagens classificadas incorretamente.

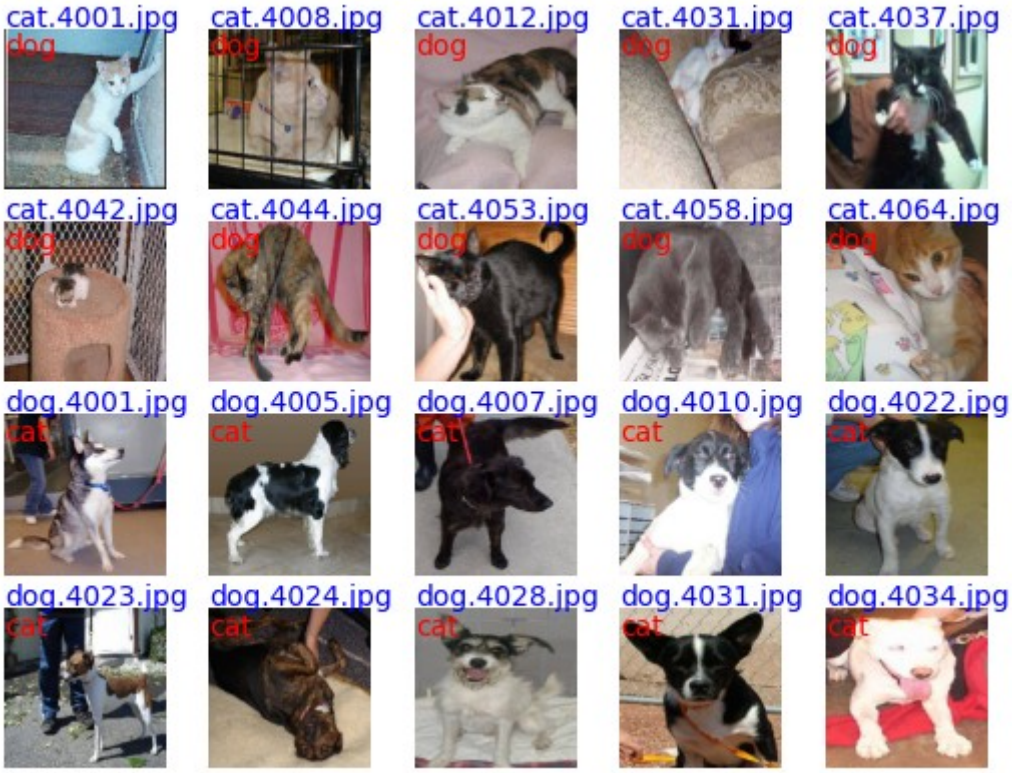

Figura 2: 10 primeiras imagens de teste classificadas incorretamente usando modelo treinado “do zero”.

2) O programa treino_transf.py deve fazer o treino usando “transfer learning”. Você deve obrigatoriamente utilizar algum modelo pré-treinado de Keras/PyTorch e adaptá-lo para o problema dado. Você deve gravar o arquivo transf.h5 (ou transf.pth) no diretório default com a rede treinada. Se quiser, pode usar “data augmentation” e outras técnicas para diminuir a taxa de erro. Depois do treino, o programa teste_transf.py deve carregar transf.h5 e classificar as imagens de teste, obtendo a taxa de erro de teste. Também deve imprimir as 10 primeiras imagens de gatos e 10 primeiras imagens de cachorros classificadas incorretamente.

Sem fazer “data augmentation” e sem fazer muito esforço para diminuir erro, obtive taxa de erro de 1,25%. A figura abaixo mostra as 10 primeiras imagens classificadas incorretamente. O que se observa é que há erros no banco de dados. A figura cat.4085.jpg claramente é imagem de um cachorro (quando deveria ser de um gato). Na imagem cat.4338.jpg, não aparece nem gato nem cachorro. A imagem cat.4522.jpg consiste de grandes áreas brancas.

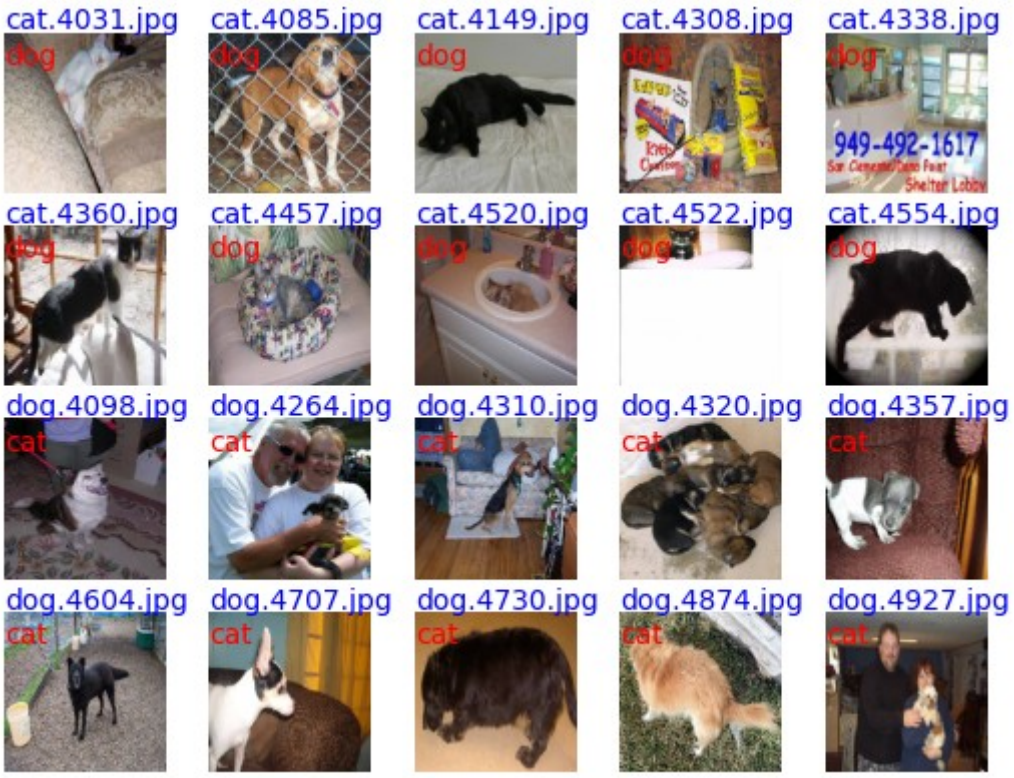

Figura 3: 10 primeiras imagens de teste classificadas incorretamente usando “transfer learning”.

Obs. 1: Entregue os programas-fontes (treino_zero.py, teste_zero.py, treino_transf.py e teste_transf.py). Você pode enviar links para notebooks Google Colab em vez de programas .py.

Obs. 2: Entregue um documento PDF de no máximo 5 páginas (relatorio.pdf) descrevendo o funcionamento dos seus programas e os resultados obtidos. O envio do relatório é obrigatório (veja o anexo).

Obs. 3: Entregue um vídeo de no máximo três minutos explicando o funcionamento dos seus programa. No vídeo deve aparecer em algum momento o seu rosto e um documento seu. De preferência envie link para vídeo e não o vídeo em si, para não sobrecarregar o meu HD.

Obs. 4: Envie os arquivos e links para hae.kim@usp.br. Dentro do prazo, você pode substituir o email anterior por um novo. Só o último email entregue será corrigido.

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 palavras simples. Use o tempo verbal o mais simples possível. 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 (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

Nome, número USP, nome da disciplina, etc.

Breve enunciado do problema

Apesar do enunciado do problema ser conhecido ao professor/monitor, descreva brevemente o problema que está resolvendo. Isto tornará o documento compreensível para alguma pessoa que não tem o enunciado do EP à mão.

Técnica(s) utilizada(s) para resolver o problema

Descreva quais técnicas você usou para resolver o problema. Se você mesmo inventou a técnica, descreva a sua ideia, deixando claro que a ideia foi sua. Se você utilizou alguma técnica já conhecida, utilize o nome próprio da técnica (por exemplo, filtragem Gaussiana, algoritmo SIFT, etc.) juntamente com alguma referência bibliográfica onde a técnica está descrita. Use elementos gráficos como imagens intermediárias e diagramas, pois ajudam muito a compreensão. Não “copie-e-cole” código-fonte, a não ser que seja relevante. Use preferencialmente o pseudo-código.

Ambiente de desenvolvimento utilizado

1 https://plainlanguage.gov/guidelines/

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

Descreva o material externo utilizado, como livros/artigos consultados, websites visitados, etc.