

Projeto de Detecção de Tubarão

Aprendizagem de Máquina 2024.2

10.04.2025

Aluno: Henrique da Silva

Aluno: Bruno França

Professor: Daniel de Filgueiras Gomes, PhD

Repositório: https://github.com/shapis/ufpe_ee/

Sumário

1	Introdução e Motivação	4
1.a	Introdução	5
1.b	Motivação	6
2	Intuição do Funcionamento	7
2.a	Ideia central	8
2.b	Execução do programa	9
3	Funcionamento do Código	10
3.a	Importação de Bibliotecas	11
3.b	Carregamento e Pré-processamento de Dados	12
3.c	Criação do Modelo	13
3.d	Treinamento com K-Fold	14
3.e	Geração de Heatmap	15
3.f	Cálculo do Centro de Massa	16
3.g	Classificação de uma Imagem	17
3.h	Geração de Vídeo com Heatmaps	18
3.i	Processamento de Imagens para Pasta Resolvida	19

Sumário (ii)

3.j FFMPEG	20
4 Resultados	21
4.a Resultados	22
4.b Resultados	23

1 Introdução e Motivação

Introdução

Este seminário apresenta um sistema de detecção de tubarões em vídeos, desenvolvido com redes neurais. O objetivo principal é classificar cenas em três categorias — “tubarão”, “mar” e “não sei” — e gerar mapas de calor para destacar as regiões com maior probabilidade de conter um tubarão.

Motivação

A detecção automática de tubarões pode contribuir para sistemas de monitoramento costeiro, oferecendo uma ferramenta auxiliar para aumentar a segurança em áreas de banho. A proposta combina visão computacional e aprendizado de máquina para oferecer uma solução acessível e eficiente.

2 Intuição do Funcionamento

Ideia central

A ideia central do programa é ensinar uma rede neural a reconhecer padrões visuais associados à presença de tubarões em imagens. Para isso, o modelo é treinado com diversos exemplos rotulados manualmente, aprendendo a diferenciar entre cenas com tubarões, apenas mar e situações ambíguas (“não sei”).

Execução do programa

Durante a execução, o vídeo é dividido em pequenos quadros (frames), e cada um deles é analisado individualmente. A rede classifica cada imagem e, além disso, gera um “heatmap” que destaca as regiões da imagem onde há maior chance de presença do tubarão. O sistema também aplica uma coloração geral ao frame: vermelha quando detecta um tubarão, azul para mar e verde para incerteza — facilitando a visualização rápida do resultado.

3 Funcionamento do Código

Importação de Bibliotecas

O código começa importando bibliotecas essenciais como os, cv2 (OpenCV), numpy, tensorflow, sklearn, matplotlib, e scipy. Elas são usadas para leitura e manipulação de imagens, construção e treinamento da rede neural, geração de gráficos e cálculos matemáticos.

Carregamento e Pré-processamento de Dados

A função **carregar_dados()** percorre as pastas de imagens organizadas por classe, redimensiona as imagens e associa os rótulos. A função **augmentar()** aplica transformações (rotação, espelhamento, ruído) para aumentar a diversidade dos dados e evitar overfitting.

Criação do Modelo

- Camada de entrada
- Camada densa com ativação ReLU
- Dropout (para regularização)
- Saída softmax com 3 neurônios (uma para cada classe)

Treinamento com K-Fold

A função **treinar_kfold()** divide os dados em 5 subconjuntos ($K=5$), treina o modelo 5 vezes e calcula a acurácia média. Isso aumenta a robustez do treinamento.

Geração de Heatmap

A função **gerar_heatmap()** percorre a imagem em blocos, prediz a probabilidade da classe “tubarão” para cada região e monta um mapa de calor que indica onde o tubarão pode estar.

Cálculo do Centro de Massa

A função **calcular_centro_massa()** usa o heatmap para encontrar a posição média (centro) das regiões com alta probabilidade (>75%).

Classificação de uma Imagem

O código faz a leitura de uma imagem de teste, prediz sua classe, calcula o heatmap e imprime o resultado com a classe mais provável.

Geração de Vídeo com Heatmaps

A função `gerar_video_heatmaps()` lê os frames de um vídeo, gera heatmaps para cada um e sobrepõe a coloração correspondente:

- Vermelho para tubarão.
- Azul para mar.
- Verde para “não sei”.

Processamento de Imagens para Pasta Resolvida

O código percorre as imagens da pasta frames, classifica cada uma e copia para frames_resolvidos com a tonalidade alterada de acordo com a classe prevista.

FFMPEG

Após o término do programa, utilizamos o FFMPEG para gerar um vídeo a partir das imagens processadas, aplicando a coloração correspondente a cada classe. O comando `ffmpeg -framerate 30 -i frames_resolvidos/%d.png -c:v libx264 -pix_fmt yuv420p video.mp4` cria um vídeo com 30 quadros por segundo.

4 Resultados

Resultados

Durante o desenvolvimento, conseguimos treinar um modelo que classifica imagens em três categorias com boa acurácia. No entanto, ao tentar localizar visualmente o tubarão por meio de um heatmap, enfrentamos dificuldades: o centro de calor nem sempre coincidiu com a real posição do tubarão no frame. Como alternativa, implementamos uma solução funcional: sobrepomos uma máscara colorida sobre o frame inteiro, de acordo com a classe prevista — vermelho para tubarão, azul para mar e verde para “não sei”.

Resultados

Esse projeto nos proporcionou aprendizados valiosos em várias frentes, como pré-processamento de dados, uso de redes neurais com softmax e regularização, técnicas de data augmentation, avaliação com K-Fold, e a aplicação prática de redes em análise de vídeo com OpenCV. Além disso, nos mostrou os desafios reais de interpretar visualmente os resultados de um modelo de visão computacional.