**

*Luciana Neves Loureiro*

*Uso de Redes Neurais Convolucionais para Detecção de Emissões Fugitivas a partir de imagens*

*Monografia de Final de Curso*

***26/06/2021***

***Monografia apresentada ao Departamento de Engenharia Elétrica da PUC/Rio como parte dos requisitos para a obtenção do título de Especialização em Business Intelligence.***

***Orientadores: Manoela Kohler***

barra direita

Dedicatória

Dedico esse estudo aos meus companheiros de trabalho que apoiaram as campanhas de monitoramento de fontes de emissões fugitivas em meio à pandemia do COVID 19.

Agradecimentos

Agradeço as professoras Carolina e Manoela pela compreensão diante dos atropelos ao longo do desenvolvimento da monografia e pelas palavras de incentivo.

# RESUMO

Esta monografia apresenta um método para detecção de emissões fugitivas (emanações de gases provenientes de componentes de linhas de processo que ocorrem, em geral, por perda de estanqueidade e de maneira randômica) de Compostos Orgânicos Voláteis (COVs) e metano (CH4), a partir de imagens geradas com câmera portátil de infravermelho (IR).

O banco de imagens adotado foi obtido através de campanhas de monitoramento realizadas em Unidades Estacionárias de Produção (UEP) de Óleo e Gás (O&G).

A detecção da presença ou ausência de vazamentos gasosos em componentes e acessórios de linhas de processo pode ser realizada a partir da avaliação “humana” de imagens e/ou vídeos gerados com câmeras IR. E o uso de Redes Neurais Convolucionais (CNNs) busca aperfeiçoar a avaliação de banco de imagens de componentes para detecção de emissões fugitivas.

# ABSTRACT

This work presents a method for detecting fugitive emissions (gas emanations from process line components that occur, in general, due to leaks in a random way) of Volatile Organic Compounds (VOCs) and methane (CH4) from images generated with a portable infrared (IR) camera.

The adopted image database was obtained through monitoring campaigns carried out in Stationary Production Units (SPU) of Oil and Gas (O&G).

The detection of the presence or absence of gas leaks in process line components and accessories is carried out based on the “human” evaluation of the images and/or videos generated with IR cameras. And the use of Convolutional Neural Networks (CNNs) seeks to improve the evaluation of the image database for detecting fugitive emissions.

**SUMÁRIO**

[RESUMO 4](#_Toc75960734)

[ABSTRACT 5](#_Toc75960735)

[1. INTRODUÇÃO 7](#_Toc75960736)

[1.1. Motivação 8](#_Toc75960737)

[1.2. Objetivos do Trabalho 9](#_Toc75960738)

[2. DESCRIÇÃO DO PROBLEMA 10](#_Toc75960739)

[3. FUNDAMENTOS TEORICOS 12](#_Toc75960740)

[3.1. Inteligência artificial, aprendizagem de máquina e aprendizagem profunda. 12](#_Toc75960741)

[3.2. *Deep Learning*: Redes Neurais Convolucionais 12](#_Toc75960742)

[4. GERAÇÃO, SELEÇÃO E SIMPLIFICAÇÃO DO BANCO DE DADOS. 15](#_Toc75960743)

[5. DISCUSSÃO DO SCRIPT E DOS RESULTADOS 17](#_Toc75960744)

[5.1. Discussão do *script* 17](#_Toc75960745)

[5.2. Resultados 18](#_Toc75960746)

[6. CONCLUSÕES E PROPOSTAS DE FUTUROS TRABALHOS 21](#_Toc75960747)

[7. REFERÊNCIAS BIBLIOGRÁFICAS 22](#_Toc75960748)

[ANEXO 1 - *Script* 24](#_Toc75960749)

# INTRODUÇÃO

Ao longo dos últimos anos, a indústria de Óleo & Gás (O&G) vem reduzido emissões de compostos orgânicos, oriundas de diversas operações da cadeia. As emissões fugitivas - emanações de gases provenientes de componentes de linhas de processo (válvulas, flanges, selos de bomba e etc.) que ocorrem, em geral, função de perda de estanqueidade e de maneira randômica, são bastante representativas nesse cenário (USEPA, 1990, Onat, 2006 e Wilwerding, 2011). Os métodos rotineiramente utilizados para detecção de emissões fugitivas são: LDAR (*Leak Detection and Repair)* - método para detecção e reparo de vazamentos definido como a principal forma adotada para identificar emissões fugitivas em componentes de linhas de processo e equipamentos através de monitoramento com uso de analisador de gases do tipo farejador ou *Sniffer*, e o OGI (*Optical Gas Imaging)* - imageamento ótico de gases que utiliza câmeras, em geral, com visualização no espectro do infravermelho (IR) para detecção de compostos orgânicos, CH4 (metano) e COVs (Compostos Orgânicos Voláteis) (Parkinson, 2007, CONCAWE, 2015 e FLIR, 2017). A possibilidade de detecção com o método OGI é dependente do tamanho do vazamento (quantidade de massa emitida) entre outros fatores. Esse método pode ser usado para detectar vazamentos gasosos em componentes de linha de processo sem grande proximidade do operador da câmera IR e devido à capacidade de detecção remota, o OGI pode encontrar vazamentos de fontes não acessíveis. A Tabela 1 apresenta algumas diferenças entre os métodos de detecção de emissões fugitivas.

**Tabela 1** – Comparativa entre os métodos para detecção de fugitivas

|  |  |  |
| --- | --- | --- |
| **MÉTODO** | **DETECÇÃO DE VAZAMENTOS (LDAR)** | **IMAGEAMENTO ÓTICO (OGI)** |
| **Instrumento de detecção** | Analisador de gases que mede a concentração na “interface” dos potenciais pontos de vazamento em campo. A sonda do analisadorprecisa estar de 1 a 2 cm de onde o vazamento ocorre. | Câmera explora as unidades de processo e detecta as plumas de emissão a longas distancias (vários metros do vazamento dependendo da sua intensidade) |
| **Aplicabilidade** | Qualquer unidade que manuseie hidrocarbonetos voláteis. Particularmente unidades onde as tubulações são acessíveis. | Unidades que manuseiem hidrocarbonetos voláteis. Particularmente, as grandes unidades ou aquelas onde existem muitos pontos potenciais de vazamento que possuem isolamento ou não são facilmente acessíveis. |
| **Resultado** | Concentração (em ppmv) na interface do componente. | Imagem onde o vazamento detectado é visualizado como uma pluma de gases. |
| **Quantificação da emissão** | Correlações entre a concentração medida em ppmv  e taxa de vazamento em Kg/h. | São aplicados fatores para componentes com /sem vazamento para todos os pontos potenciais. |

**Fonte**: Elaborado a partir de Siegell, 1997, Parkinson, 2007 e CONCAWE, 2015.

### Motivação

A gestão de emissões fugitivas é laboriosa e de alto custo, dessa forma é importante buscar alternativas para torna-la mais custo efetiva (USEPA, 2011 e CONCAWE, 2017). E uma das linhas mais promissoras é a utilização de ferramentas para otimização do uso do método OGI, que ainda demanda muita atuação humana, apesar dos crescentes desenvolvimentos em quantificação dos vazamentos a partir de imagens - qOGI (*quantitative Optical Gas Imaging*) - imageamento ótico de gases quantitativo, que adota técnicas de modelagem matemática que utilizam dados complementares (como distância da câmera para fonte de vazamento, condições ambientais (direção e velocidade do vento e propriedades óticas dos gases) para calcular parâmetros de emissão em termos de massa (CONCAWE, 2017).

Dessa forma, métodos eficientes para gestão de emissões fugitivas com a possibilidade de automatização da análise para identificação de componentes com vazamentos gasosos em dezenas de horas de vídeos e/ou centenas de fotos são prementes.

Esse trabalho é uma proposta de utilização de redes neurais convolucionais (CNN) para o processamento de informações visuais de componentes para detecção de emissões fugitivas. Ou seja, análise das imagens de componentes de linha de processo para identificação de quais estão com ou sem vazamento gasoso.

### Objetivos do Trabalho

O desenvolvimento dessa monografia envolveu seis etapas descritas nos seguintes itens:

* 2 - Descrição do problema
* 3 - Fundamentos teóricos
* 4 - Geração, seleção e simplificação do banco de dados
* 5 - Discussão do script e dos resultados
* 6 – Conclusões
* 7 - Bibliografia

O item 2 apresenta uma descrição sobre emissões fugitivas e uma panorâmica sobre questões vinculadas ao tema. O item 3 contem os fundamentos teóricos sobre inteligência artificial, aprendizagem de máquina e aprendizagem profunda e detalhamento sobre *Deep Learning*: Redes Neurais Convolucionais O item 4 descreve como foi elaborado o banco de dados, os critérios adotados para seleção de dados e as simplificações assumidas. Já no item 5 é apresentada uma discussão sobre o script adotado e os resultados obtidos. O item 6 descreve as conclusões e propõem possíveis trabalhos futuros. E finalmente no item 7 é apresentada a bibliografia adotada para desenvolvimento dessa monografia.

# DESCRIÇÃO DO PROBLEMA

Emissões fugitivas são emanações de gases provenientes de componentes de linhas de processo (válvulas, flanges, selos de bombas, etc.) que ocorrem, em geral, por perda de estanqueidade e de maneira randômica. Uma unidade industrial possui milhares de componentes passíveis de avaliação em termos de emissões fugitivas. A identificação dos componentes com vazamento em uma unidade é uma tarefa demorada, uma vez que esses pontos estão espalhados, não são visíveis a olho nu e por isso demandam o uso de métodos para detecção (USEPA, 2011 e CONCAWE, 2017). Tradicionalmente, existem dois métodos para detectar vazamentos gasosos em componentes: a) LDAR e o OGI. A aplicação do método OGI utiliza câmeras IR para detectar vazamentos. Essas câmeras são equipadas com filtro para detectar seletivamente a radiação na banda de absorção específica de certos compostos orgânicos. Compostos orgânicos alifáticos e aromáticos podem absorver radiação IR na faixa espectral de 3,2 - 3,4 μm. A câmera IR, GF320 da FLIR, foi adotada para geração de imagens utilizadas nesse estudo. A Tabela 2 mostra os compostos orgânicos que absorvem luz nessa estreita faixa do espectro e que são passiveis de serem detectados pela GF320.

**Tabela 2** – Compostos orgânicos que a câmera GF 320 - FLIR pode detectar.

|  |  |
| --- | --- |
| **NOME DO COMPOSTO ORGÂNICO** | |
| Metano | Etanol |
| Etano | Eteno |
| Propano | Propeno |
| Butano | Penteno |
| Pentano | Metil etil cetona (MEK) |
| Hexano | Metil isobutil cetona (MIBK) |
| Heptano | Benzeno |
| Octano | Tolueno |
| Isopreno | Xilenos (orto,meta e para ) |
| Metanol | Etil benzeno (BTEX) |

**Fonte:** Elaboração própria a partir de FLIR, 2017.

Basicamente, as câmeras IR são compostas de:

- Lente;

- Filtro;

- Detector;

- Eletrônicos (para o processamento do sinal do detector);

- Visor.

As emissões de compostos orgânicos podem ser visualizadas nas câmeras IR, caso três condições sejam atendidas:

- O composto orgânico só é detectado se o seu espectro de absorção no infravermelho se sobrepõe à banda de absorção adequada para o filtro da câmera (banda estreita);

- Deve existir uma diferença de temperatura entre a radiação de fundo emitida e o total de radiação emitida pelos compostos orgânicos;

- A pluma deve estar em movimento.

Na tela da câmera IR, a pluma de poluentes do vazamento é vista como uma imagem, que deve ser registrada como vídeo ou como foto.

Em geral, uma câmera de IR tem várias formas de apresentar uma imagem, os modelos mais recentes possuem o modo de alta sensibilidade que permite que pequenas quantidades de compostos orgânicos fiquem visíveis durante o uso da câmera. No modo de alta sensibilidade (HSM-*High Sensivity Mode*) as imagens consecutivas são subtraídas entre si de forma a acentuar o movimento de emissões. Durante a operação de detecção de vazamentos a câmera IR deve ser posicionada em diversos ângulos, e utilizada nos variados modos de operação. As figuras 1 (a), (b) e (c) mostram imagens de componente de linha (cap da válvula)

|  |  |  |
| --- | --- | --- |
| (a) | (b) | (c) |

**Figura 1** – (a) Visualização de válvula a olho nu, vazamento não pode ser visto; (b) Visualização de vazamento no cap da válvula com câmera IR, modo automático; (c) Vazamento no cap da válvula com câmera IR, no modo de alta sensibilidade.

Na tela da câmera, a pluma de composto(s) orgânico(s) pode ser visualizada (o que não pode ser visto a olho nu). Cada vazamento identificado (detectado) pode ser registrado na forma de vídeo e/ou foto para posteriores ações (registro e solicitação de reparo).

# FUNDAMENTOS TEORICOS

### 3.1. Inteligência artificial, aprendizagem de máquina e aprendizagem profunda.

Segundo Jia *et a*l, 2021, Gulli *et al*, 2017, Chollett, 2018 e Dong *et al,* 2021 a Inteligência Artificial (IA) é um campo de pesquisa amplo, em que máquinas apresentam recursos como: aprendizagem, interação proativa com o ambiente, inferência, dedução, visão computacional, reconhecimento de fala, resolução de problemas, representação de conhecimento, percepção, entre muitos outros. De maneira ampla, IA denota qualquer atividade em que as máquinas imitem comportamentos tipicamente humanos. A IA se inspira em elementos de ciência da computação, matemática e estatística

O aprendizado de máquina (*machine learning*– ML) é uma sub-ramificação da IA que se concentra em ensinar computadores a aprender sem a necessidade do mesmo ser programado para tarefas específicas. A ideia-chave do ML é criar algoritmos que aprendam e façam previsões sobre um conjunto de dados. Existem três categorias de ML:

* aprendizagem supervisionada, a máquina é apresentada a entradas dados e saída desejada, e o objetivo é aprender com esses exemplos de treinamento de forma que previsões significativas podem ser feitas para novos dados nunca antes vistos.
* aprendizagem não supervisionada, a máquina é apresentada a dados de entrada apenas e tem que encontrar alguma estrutura significativa por si só, sem supervisão externa.
* Na aprendizagem por reforço, a máquina atua como um agente, interagindo com o ambiente e aprendendo quais são os comportamentos que geram recompensas.

Aprendizagem profunda (*Deep Learning* - DL) é um subconjunto específico de metodologias de ML que usam redes neurais artificiais (*Artificial Neural Net* - ANN) inspirados na estrutura dos neurônios localizados no cérebro humano. Informalmente, a palavra profunda se refere à presença de muitas camadas na rede neural artificial.

### 3.2. *Deep Learning*: Redes Neurais Convolucionais

Redes neurais artificiais é uma classe de modelos de aprendizado de máquina, inspirada em estudos sobre o sistema nervoso central de mamíferos. Cada rede é composta por vários neurônios interconectados, organizados em camadas, que trocam informações quando certas condições acontecem (Jia *et a*l, 2021, Gulli *et al*, 2017, Chollett, 2018 e Dong *et al,* 2021).

As redes neurais convolucionais (*Convolution Neural Net* - CNN) são provavelmente o modelo de rede *DL* (aprendizado com representações hierárquicas[[1]](#footnote-1)) mais conhecidas e utilizadas atualmente devido às aplicações com sucesso no processamento e análise de imagens digitais. A aplicação das CNNs para o processamento de informações visuais, em particular imagens, ocorre porque a convolução permite filtrar as imagens considerando-as com uma estrutura bidimensional.

As CNNs são compostas basicamente de camadas convolucionais, que processam as entradas considerando campos receptivos locais. A convolução é uma operação matemática entre duas funções “f” e “g”, que gera uma terceira função modificada de f. Nas CNNs cada neurônio se encarrega de uma parte da matriz e realiza uma convolução que extrai características em múltiplos níveis de abstração, proporcionando melhores analises quando se tem múltiplas variáveis (Zhao *et al*, 2019, Kolar *et al* ,2018 e Dong *et al*, 2021).

Na camada convolucional, cada neurônio é um filtro aplicado a uma imagem de entrada, nessa camada a imagem é processada e transformada por meio de uma combinação linear dos pixels vizinhos. Nesse processamento, informações locais são utilizadas para cada posição (x, y) tomada como centro, essa região é conhecida como campo receptivo. E seus valores são utilizados como entrada para um filtro *i* com parâmetros *wi*, produzindo um único valor (pixel) no mapa de características f(i, x, y) da saída de cada neurônio da camada convolucional. (Kolar *et al* ,2018 e Dong *et al*, 2021).

Todos os campos receptivos são filtrados com os mesmos pesos locais para todos os pixels. Assim, um valor de saída ainda terá o formato bidimensional. No processamento de imagens, onde a imagem é definida como uma função bidimensional, a convolução é útil para detecção de bordas, suavização de imagem, extração de atributos, entre outras aplicações.

Em linhas gerais o processo de utilização das CNNs prevê os seguintes passos (Kolar *et al* ,2018, Chollett, 2018 e Dong *et al,* 2021):

1 - Escolha do filtro(*Kernel*) – dimensão

2 – Determinação do passo (*Stride*) – é um parâmetro da camada convolucional usado para diminuir o volume de informações 3D. O algoritmo só processa Enésima linha e coluna. Se o *stride* for igual a 1, o filtro move um pixel de uma vez. Se volumes de saída menores forem desejados, um *stride* de 2 pode ser escolhido, o que leva em consideração apenas cada segunda linha e coluna.

3 - Uso de preenchimento (*Padding)* - técnica de preenchimento adotada quando a convolução é realizada perto das bordas de uma imagem, há informação ausente fora da imagem e para que o algoritmo processe as informações, é necessário adicionar (0,0,0) ou alguma outra forma de preenchimento (como por exemplo, copiar, espelhar as informações ao longo das bordas).

4 – Uso de *pooling* – operação de redução da dimensionalidade espacial das representações, na qual as dimensões X e Y do volume 3D são reduzidas. As camadas de agrupamento são removidas em cada enésima linha e coluna de um volume 3D.

5 - Uso de *Flatening* - camada normalmente utilizada para divisão da CNN em 2 partes (de extração de características e rede neural tradicional). Basicamente transforma a matriz imagem, em formato de *Array* (estrutura de dados que segue uma sequência não ordenada)

6 – Uso de *Dropout* - processo de eliminação (temporária) de alguns dos neurônios na rede, contudo deixando os neurônios de entrada e saída intocados.

7 – Uso de função de ativação – utilização de uma função que traga não-linearidades ao sistema, de forma que a rede consiga aprender qualquer tipo de funcionalidade. Em CNNs comumente se adota a função ReLu (*Rectified Linear Units*) por ser mais eficiente computacionalmente sem grandes diferenças de acurácia quando comparada a outras funções. A ReLu zera todos os valores negativos da saída da camada anterior.

8 – Uso de camada *Fully Connected* (FC) ou totalmente conectada – camada cuja entrada é a saída da camada anterior e sua saída é “N” neurônios, com ”N” sendo a quantidade de classes do modelo para finalizar a classificação. Ou seja, no final da rede, o volume 3D de neurônios é convertido em camadas totalmente conectadas, que determinam a classe da imagem na camada de saída.

# GERAÇÃO, SELEÇÃO E SIMPLIFICAÇÃO DO BANCO DE DADOS.

O banco de dados utilizado no *script* foi gerado a partir de campanhas de monitoramento para detecção de emissões fugitivas em unidades de produção de O&G em ambiente offshore. Durante o monitoramento, a câmera IR foi posicionada em diversos ângulos de visão. A pluma identificada foi registrada como vídeo ou foto para avaliações subsequentes.

Os principais fatores que afetam a detecção de vazamentos pelo método OGI são: parâmetros meteorológicos (ex: temperatura, vento, umidade, radiação solar), treinamento do operador da câmera, tempo de observação dos componentes, ângulos e distâncias de visualização, bem como os tipos de compostos presentes na emissão gasosa. Quando a câmera IR está em operação, a mesma detecta a absorção de luz infravermelha de uma pluma de gás. O fluxo de radiação infravermelha (IR) está condicionado à diferença de temperatura. Para que uma imagem seja gravada pela câmera IR, deve haver uma diferença de temperatura (∆T) entre o gás que está sendo emitido e o fundo (*background*). O *database* utilizado para processamento foi obtido via extração de fotos a partir de vídeos realizados com a câmera IR GF-320 (FLIR). Os componentes de linha de processo selecionados foram os caps de válvulas e flanges, a Tabela 3 apresenta o conteúdo do banco de dados, binários: COMfugitivas e SEMfugitivas.

**Tabela 3** – Detalhamento do banco de dados

|  |  |  |  |
| --- | --- | --- | --- |
| **COMfugitivas** | | **SEMfugitivas** | |
| 1\_CAP.jpg | FLANGE\_3.jpg | CAP\_S\_1.jpg | FLANGE\_SEM\_8.jpg |
| 2\_CAP.jpg | FLANGE\_4.jpg | CAP\_S\_2.jpg | FLANGE\_SEM\_9.jpg |
| 3\_CAP.jpg | FLANGE\_5.jpg | CAP\_S\_3.jpg | FLANGE\_SEM\_10.jpg |
| 4\_CAP.jpg | FLANGE\_6.jpg | FLANGE\_SEM\_1.jpg | FLANGE\_SEM\_11.jpg |
| 5\_CAP.jpg | FLANGE\_7.jpg | FLANGE\_SEM\_2.jpg | FLANGE\_SEM\_12.jpg |
| 6\_CAP.jpg | FLANGE\_8.jpg | FLANGE\_SEM\_3.jpg | FLANGE\_SEM\_13.jpg |
| 7\_CAP.jpg | FLANGE\_9.jpg | FLANGE\_SEM\_4.jpg | FLANGE\_SEM\_14.jpg |
| 8\_CAP.jpg | FLANGE\_10.jpg | FLANGE\_SEM\_5.jpg |  |
| FLANGE\_1.jpg | FLANGE\_11.jpg | FLANGE\_SEM\_6.jpg |  |
| FLANGE\_2.jpg | FLANGE\_12.jpg | FLANGE\_SEM\_7.jpg |  |

As imagens geradas foram salvas em *Joint Photographics Experts Group* (JPEG) que é um dos formatos mais populares, devido à boa taxa de compactação em 24 bits (16 milhões de cores), e permitir a escolher a taxa de compactação dos dados da imagem (quanto mais compactado menor a qualidade). Os padrões de compactação de imagens tradicionais, como JPEG ou JPG, são construídos sobre a transformação e quantização em bloco, cujo objetivo é reduzir o número de bits de codificação (Zhao *et al*, 2019).

# 5. DISCUSSÃO DO SCRIPT E DOS RESULTADOS

### 5.1. Discussão do *script*

O *script* utilizado (ver anexo 1 – realizado no Jupyter Notebook on Google Colab) foi escrito na linguagem de programação Python, que possui muitas bibliotecas disponíveis em código aberto (como o Tensoflow)

O script adotou o API (*Application Programming Interface*) do Tensorflow, KERAS, um modelo sequencial do tipo *pipeline* com camadas de redes neurais.

O conjunto de dados foi dividido em dois: treinamento e teste. O conjunto de dados adotado (*dataset*) está descrito na Tabela 3.

A fim de melhorar a capacidade de generalização do classificador a partir do conjunto de dados e aumentar o *dataset* foram utilizadas as operações de Data *Augmentation e ImageDataGenerator*.

Para melhorar a capacidade de generalização do classificador e aumentar a quantidade de dados, técnicas de transformação (por exemplo, rotação, alongamento, inversão, etc.) foram utilizadas em cada imagem com componentes de linhas de processo, dessa forma a quantidade de imagens foi aumentada.

Outra estratégia adotada foi o uso de modelo pré-treinado para aprendizado das características das imagens que foram utilizadas a fim de diferenciar uma imagem de outra (classes).

Nesse script foi utilizado o modelo pré-treinado, VGG-16, que transferiu o conhecimento sobre características de imagens armazenadas nesse modelo. A razão de usar o VGG-16 é que o mesmo foi treinado com o banco de dados *ImageNet* que contém 1000 classes. Isso significa que as camadas convolucionais têm a capacidade de generalizar suficientemente a fim de reconhecer características grosseiras de muitos objetos diferentes, que variam em forma, cor, textura, etc. Portanto, o VGG-16 tem um bom desempenho como extrator de recursos para o *dataset* adotado. O 16 no VGG16 refere-se as suas 16 camadas que possuem pesos.

O número de *epochs* ou épocas foi gradualmente aumentado durante os treinos: 5, 10, 15, 20, 50 e adotou-se 200. O que significa que o modelo parará de treinar se não houver aumento na precisão da validação em 200 épocas.

O modelo foi treinado utilizando o otimizador Adam. Os hiperparâmetros b1 e b2 no otimizador Adam controlam as taxas de decaimento exponencial das médias móveis, que influenciam o desempenho da estabilidade do treinamento.

A cada interação o otimizador Adam tentou ajustar os pesos com o objetivo de minimização. E a acurácia foi a métrica para avaliar as predições.

A função Softmax foi usada para calcular a probabilidade da categoria, onde K é o número de categorias de imagens nesse script K é igual a 2 (com fugitiva, sem fugitiva). Dessa forma a camada Softmax dá dois valores de probabilidade Z = [z1, z2]. Todas as camadas convolucionais são seguidas por uma camada de ativação com função ReLU, exceto a última camada convolucional.

Foi construída uma camada FC, que consiste em camadas POOL => FC => SOFTMAX e anexada no topo do VGG16. Em seguida, os pesos foram congelados (CONV da VGG16) de modo que apenas a cabeça da camada FC seja treinada; isso completa a configuração de ajuste fino.

### 5.2. Resultados

As métricas calculada durante os experimentos foram a acurácia (*accuracy*) e as perdas (*loss*). Os resultados obtidos demonstram que foi viável resolver o problema de detecção de emissões fugitivas por imagens com aprendizagem de transferência de CNN pré-treinado com imagens reais. A tabela 4 apresenta os principais resultados obtidos:

**Tabela 4** – Detalhamento dos resultados 1

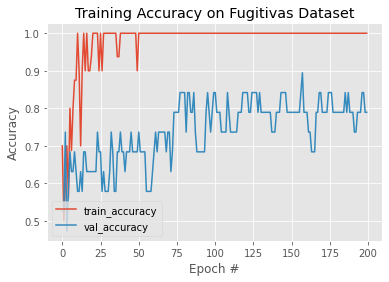
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **recall** | **f1-score** | **support** |
| **COMfugitivas** | 0,90 | 0.90 | 0.90 | 10 |
| **SEMfugitivas** | 0,89 | 0,89 | 0,89 | 9 |
| **accuracy** |  |  | 0.89 | 19 |
| **macro avg** | 0,89 | 0.89 | 0.89 | 19 |
| **weighted avg** | 0,63 | 0.62 | 0.62 | 19 |
|  | | | | |
| **accuracy** | 0,8947 | | | |
| **sensitivity:** | 0,9000 | | | |
| **specificity** | 0,8889 | | | |

A acurácia foi de 89,47%, a sensitividade de 90% e a especificidade de 88,9% significa:

Das imagens em que existem emissões fugitivas (i.e, positivos verdadeiros), foi possível identificar corretamente “COMFugitivas”, 90% do tempo utilizando o modelo. Das imagens em que não existem emissões fugitivas (i.e, negativos verdadeiros), foi possível identificar corretamente “SEMFugitivas”, 88,9% do tempo utilizando o modelo.

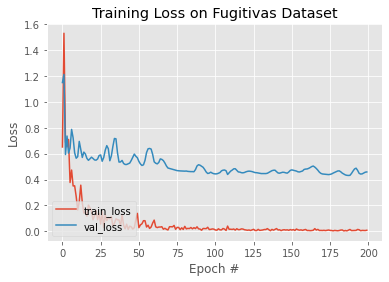
A Figura 2 histórico de treinamento da rede para a métrica acurácia

Acurácia



**Figura 2** – Acurácia X Épocas

A Figura 3 histórico de treinamento da rede para a métrica perdas



**Figura 3** – Perdas X Épocas

Os resultados das métricas obtidos atestam que a quantidade de imagens utilizadas no *database* é que propicia a robustez na validação. E que quanto mais o modelo treina maior se torna a sua acuraria diminuindo as perdas. Também cabe lembrar que se tratam de imagens REAIS de diferentes lugares e que a falta de uniformidade afeta a desempenho do sistema

No capítulo 6 a seguir são propostas estratégias a serem adotadas em trabalhos futuros.

# 6. CONCLUSÕES E PROPOSTAS DE FUTUROS TRABALHOS

A CNN gerada foi capaz de extrair características representativas da base dados. Apesar da pluma de emissão nos variados cenários apresenta diferentes características de visão. Uma possível solução para melhorar a detecção é adotar um conjunto maior de dados e principalmente com uniformidade de tomada de imagens para treinar de maneira mais satisfatória o modelo de reconhecimento de pluma a fim de resolver o problema.

O estudo apresentado demonstrou que o uso de DL pode localizar e reconhecer a pluma, com algum conhecimento prévio, como movimento e cor, combinando com recursos de baixo nível e recursos abstratos profundos propiciando construir uma representação da presença de emissão fugitiva em componentes de linhas de processo.

Outras redes de TL diferentes da VGG16 podem ser testadas como a Inception e a ResNet

A base de dados foi pequena apesar do uso da técnica de Data Augmentation com rotação de 30 °, dessa forma em futuras simulações a adoção de outros recursos como cisalhamento ou reordenação da dimensão possa incluída.

Também se aconselha que novos trabalhos devam adotar fotos diretamente geradas pela câmera IR e não uso de captura a partir dos vídeos. Ou até mesmo a utilização de vídeos.

# 7. REFERÊNCIAS BIBLIOGRÁFICAS

1. Chollet, F, Deep Learning with Python, New York, 2018
2. CONCAWE, Techniques for detecting and quantifying fugitive emisions – Results of comparative field studies, Report no. 6/15, 2015 (Disponível em: https://www.concawe.eu)
3. CONCAWE, An evaluation of an optical gas imaging system for the quantification of fugitive hydrocarbon emissions, report no. 2/17, 2017 (Disponível em: https://www.concawe.eu)
4. FLIR. Disponível em: <http://www.flir.com.br/ogi/display/?id=55671>, 2017.
5. Ian Goodfellow, I., Bengio Y., Courville A, Deep Learning
6. Gulli, A. e Pal S, Deep Learning with Keras. Packt, Birmingham, Mumbai. 2017.
7. ONAT, A., A review of fugitive emissions. Sealing Technology, p.7-9, 2006.
8. PARKINSON, Gerald. A Smarter Way to Detect Fugitive Emissions. Chemical Engineering Progress, v.103, n.12, p.7-11, 2007.
9. SIEGELL, J. H, Control valve fugitive emissions. Hydrocarbon Processing, p.45-47, 1997.
10. USEPA, Determination of Volatile Organic Compound Leaks - Method 21, Code of Federal Regulations, Title 40, Part 60, Appendix A, 40 CFR 60 Appendix A, Section 8.1.1.1, 1990.
11. USEPA, Alternative work practice to detect leaks from equipment: final rule. EPA-HQ[1]OAR-2003-0199, 2008
12. USEPA, Oil and Natural Gas Sector: Standards of Performance for Crude Oil and Natural Gas Production, Transmission and Distribution. EPA-453/R-11-002, 2011.
13. WILWERDING, J. Fugitive emissions from valves: Update. Hydrocarbon Processing, 2011. (Disponível em: [www.hydrocarbonprocessing.com/magazine/2011](http://www.hydrocarbonprocessing.com/magazine/2011)).
14. Y. Jia, W. Chen, M. Yang, L. Wang, D. Liu, Q. Zhang, Video smoke detection with domain knowledge and transfer learning from deep convolutional neural networks, Optik 240, 2021.
15. L. Zhao, H. Bai, J. Liang, B. A. Wang, Y. Zhao, Learning a virtual codec based on deep convolutional neural network to compress image, J. Vis. Commun. Image R. 63, 2019.
16. Z. Kolar , X. Ren, H. Chen, X. Luo, Transfer learning and deep convolutional neural networks for safety guardrail detection in 2D images, Autom. Constr. 89, 2018.
17. S. Dong, P. Wang, K. Abbas, A survey on deep learning and its applications, Computer Science Review 40, 2021.

# ANEXO 1 - *Script*

Utilizando Jupyter Notebook on Google Colab

#Detecção de FUGITIVAS in images with Keras, TensorFlow, and Deep Learning

# Importando as bibliotecas necessárias

Import sys

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import VGG16

from tensorflow.keras.layers import AveragePooling2D

from tensorflow.keras.layers import Dropout

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Input

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.utils import to\_categorical

from sklearn.preprocessing import LabelBinarizer

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from imutils import paths

import matplotlib.pyplot as plt

import numpy as np

import argparse

import cv2

import os

import glob

from imutils import paths

# initial learning rate, number of epochs to train for,

# and batch size

INIT\_LR = 1e-3

EPOCHS = 200

BS = 8

# grab the list of images in our dataset directory, then initialize

# the list of data (i.e., images) and class images

dataset\_path = ‘Fugitivas’

print("[INFO] loading images...")

imagePaths = list(paths.list\_images(dataset\_path))

data = [ ]

labels =[ ]

# loop over the image paths

for imagePath in imagePaths:

# extract the class label from the filename

label = imagePath.split(os.path.sep)[-2]

# load the image, swap color channels, and resize it to be a fixed

# 224x224 pixels while ignoring aspect ratio

image = cv2.imread(imagePath)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

image = cv2.resize(image, (224, 224))

# update the data and labels lists, respectively

data.append(image)

labels.append(label)

print(‘labels: “, np.unique(labels))

labels: [‘ COMfugitivas’ ‘SEMfugitivas’]

# convert the data and labels to NumPy arrays while scaling the pixel

# intensities to the range [0, 1]

data = np.array(data) / 255.0

labels = np.array(labels)

plt.figure(figsize=(10,10))

plt.imshow(data[labels==’SEMfugitivas’][3])

plt.show()

# perform one-hot encoding on the labels

lb = LabelBinarizer()

labels = lb.fit\_transform(labels)

labels = to\_categorical(labels)

# partition the data into training and testing splits using 80% of

# the data for training and the remaining 20% for testing

(trainX, testX, trainY, testY) = train\_test\_split(data, labels,

test\_size=0.50,

stratify=labels,

random\_state=42)

# initialize the training data augmentation object

trainAug = ImageDataGenerator(rotation\_range=30,fill\_mode="nearest")

# load the VGG16 network, ensuring the head FC layer sets are left

# off

baseModel = VGG16(weights="imagenet", include\_top=False,

input\_tensor=Input(shape=(224, 224, 3)))

# marcar camadas como não treináveis

for layer in baseModel.layers:

layer.trainable = False

# adicionar novas camadas para classificação

flat1 = Flatten()(baseModel.layers[-1].output)

# class1 = Dense(128, activation='relu', kernel\_initializer='he\_uniform')(flat1)

output = Dense(2, activation='softmax')(flat1)

# definir o modelo

model = Model(inputs=baseModel.inputs, outputs=output)

# compilar modelo

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Visualizar o modelo compilado

model.summary()

baseModel = VGG16(weights="imagenet", include\_top=False,

input\_tensor=Input(shape=(224, 224, 3)))

f1 = baseModel.layers[1].output

f2 = baseModel.layers[2].output

f3 = baseModel.layers[3].output

f4 = baseModel.layers[4].output

feature\_maps = Model(inputs=baseModel,input, output=[f1,f2,f3,f4])

y = np.argmac(trainY, axis=1)

feat1, feat2, feat3, feat4 = feature\_maps.predict(trainX[y==1][0:1])

fig, axs = plt.subplots(2, 2, figsize=(10, 10))

axs[0, 0].imshow(feat1[0, :, :, 0:3])

axs[0, 1].imshow(feat2[0, :, :, 6:9])

axs[1, 0].imshow(feat3[0, :, :, 0:3])

axs[1, 1].imshow(feat4[0, :, :, 3:6])

# plt.subplots

# plt.imshow(feat1[0, :, :, 0:3])

plt.show()

# train the head of the network

from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint('best4.h5', monitor='val\_accuracy', save\_best\_only=True)

print("[INFO] training head...")

H = model.fit\_generator(

trainAug.flow(trainX, trainY, batch\_size=BS),

steps\_per\_epoch=len(trainX) // BS,

validation\_data=(testX, testY),

validation\_steps=len(testX) // BS,

epochs=EPOCHS,

callbacks=[checkpoint])

model.load\_weights("best4.h5")

# make predictions on the testing set

print("[INFO] evaluating network...")

predIdxs = model.predict(testX, batch\_size=BS)

# for each image in the testing set we need to find the index of the

# label with corresponding largest predicted probability

predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report

print(classification\_report(testY.argmax(axis=1), predIdxs,

target\_names=lb.classes\_))

# compute the confusion matrix and and use it to derive the raw

# accuracy, sensitivity, and specificity

cm = confusion\_matrix(testY.argmax(axis=1), predIdxs)

total = sum(sum(cm))

accuracy = (cm[0, 0] + cm[1, 1]) / total

sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])

specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])

# show the confusion matrix, accuracy, sensitivity, and specificity

print(cm)

print("accuracy: {:.4f}".format(accuracy))

print("sensitivity: {:.4f}".format(sensitivity))

print("specificity: {:.4f}".format(specificity))

# plot the training loss and accuracy

N = EPOCHS

plt.style.use("ggplot")

plt.figure()

plt.plot(np.arange(0, N), H.history["loss"], label="train\_loss")

plt.plot(np.arange(0, N), H.history["val\_loss"], label="val\_loss")

plt.title("Training Loss on Fugitivas Dataset")

plt.xlabel("Epoch #")

plt.ylabel("Loss")

plt.legend(loc="lower left")

plt.show()

# plot the training loss and accuracy

N = EPOCHS

plt.style.use("ggplot")

plt.figure()

plt.plot(np.arange(0, N), H.history["accuracy"], label="train\_accuracy")

plt.plot(np.arange(0, N), H.history["val\_accuracy"], label="val\_accuracy")

plt.title("Training Accuracy on Fugitivas Dataset")

plt.xlabel("Epoch #")

plt.ylabel("Accuracy")

plt.legend(loc="lower left")

plt.show()

1. Hierarquia - com nível crescente de abstração [↑](#footnote-ref-1)