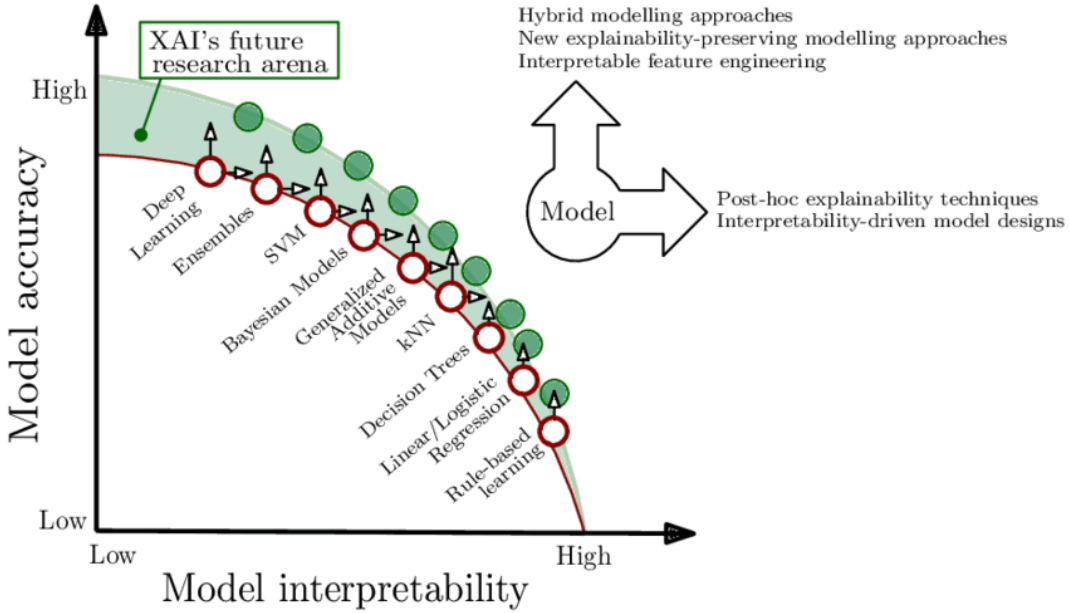
Visão Computacional

Class Activation Map (CAM)

Prof. Dr. Denis Mayr Lima Martins

Pontifícia Universidade Católica de Campinas

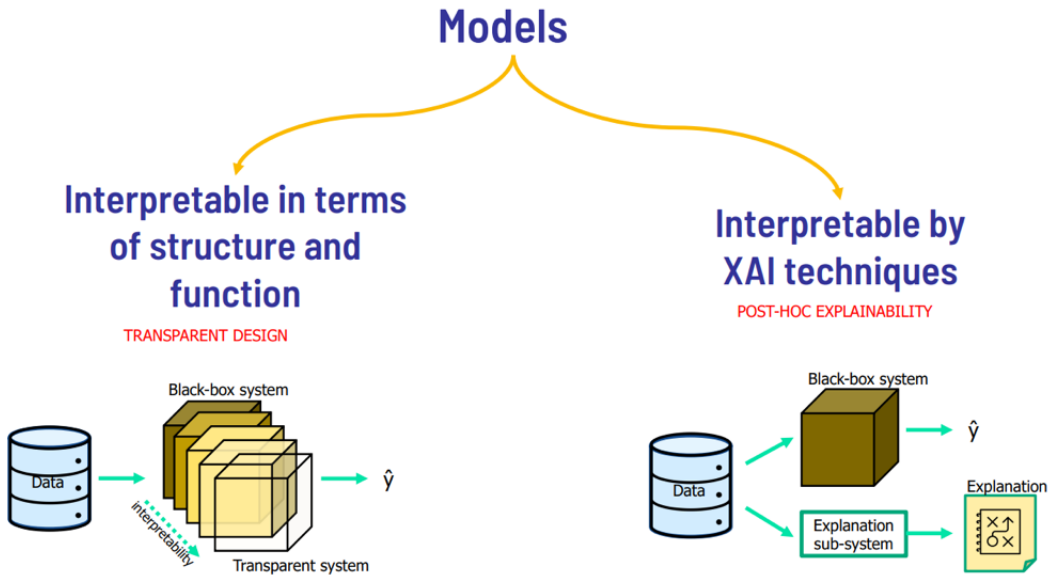
Modelos Opacos x Modelos Transparentes



Trade-off interpretabilidade versus capacidade do modelo. Fonte: Arrieta, Alejandro Barredo, et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI." Information fusion 58 (2020): 82-115.

A Necessidade de Explicabilidade

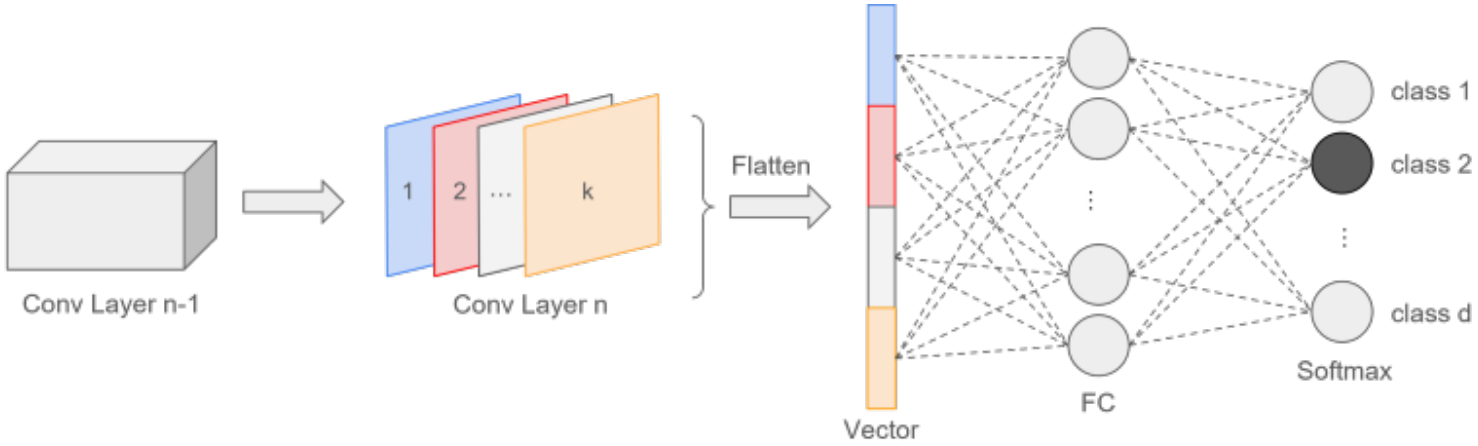
- **CNNs como "Caixas Pretas"**: Modelos de Deep Learning alcançam alta performance, mas suas previsões são difíceis de interpretar.
- **Interpretabilidade e Explicabilidade (XAI)**: O campo de Explainable AI (XAI) busca desvendar o processo de tomada de decisão desses modelos complexos.
- **Confiança e Transparência**: A falta de transparência reduz a confiança, sendo crucial verificar se o modelo está focando nas regiões corretas da imagem.



Interpretabilidade em Modelos. Fonte: Deep Learning of Python.

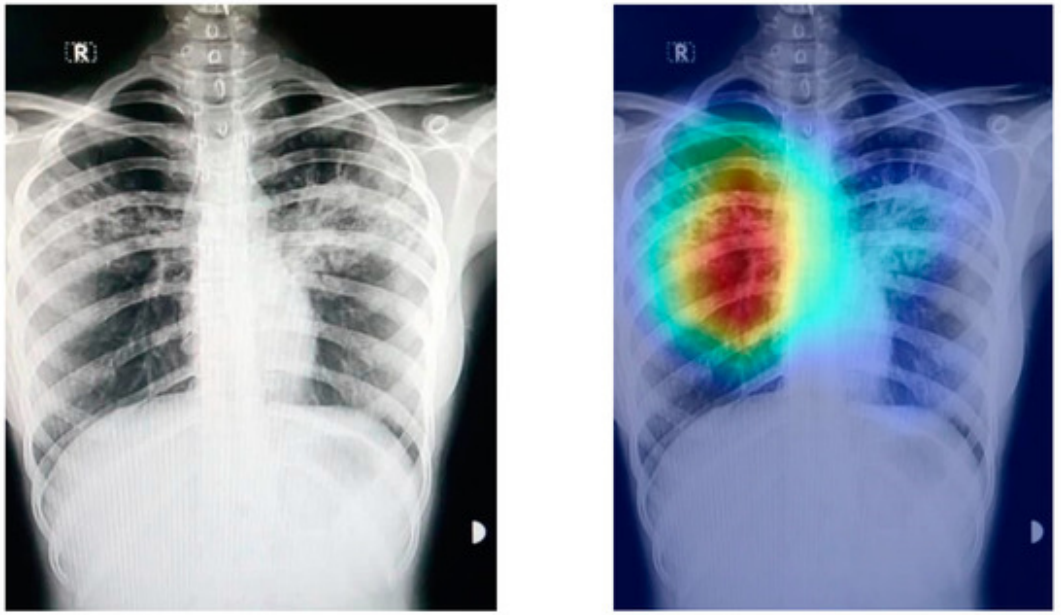
Falta de Interpretabilidade em CNNs

- **Perda da Correspondência Espacial**: O uso tradicional de camadas totalmente conectadas (FC) após as convoluções "achata" os mapas de características, agindo como uma caixa preta e perdendo a correspondência direta entre a localização espacial da característica e o *output* final.
- **Localização Discriminativa**: A capacidade da CNN de localizar objetos (inerente às camadas convolucionais) é perdida antes da classificação final.



Perda de localização por "achatamento" para camadas densas. Fonte: Joh Fischer.

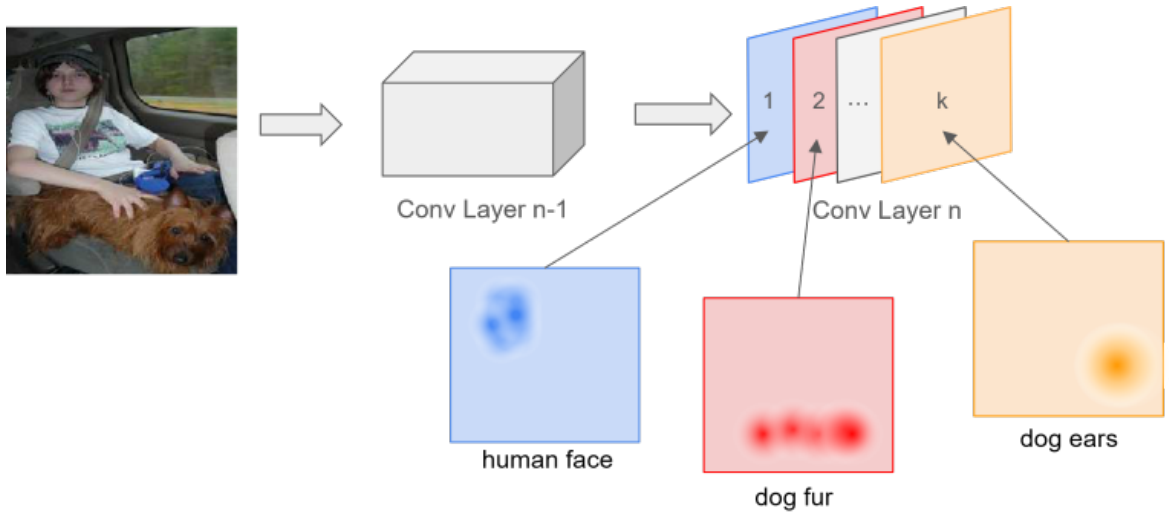
A Necessidade de Explicabilidade



Interpretabilidade em CNNs. Fonte: Umair, Muhammad, et al. "Detection of COVID-19 using transfer learning and Grad-CAM visualization on indigenously collected X-ray dataset." Sensors 21.17 (2021): 5813.

Falta de Interpretabilidade em CNNs

- **Filtros de Camadas Iniciais**: Filtros próximos à entrada detectam características de baixo nível (bordas, linhas).
- **Filtros de Camadas Profundas**: Em camadas mais profundas, as *feature maps* (mapas de características) combinam padrões, podendo corresponder a objetos ou conceitos.



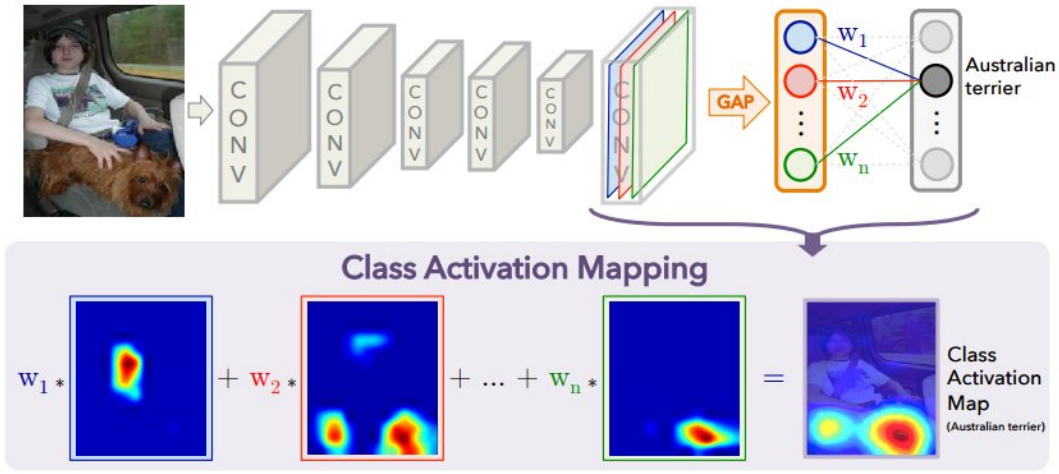
Exemplo de ativações de uma rede que reconhece pessoas e cães. Fonte: Joh Fischer.

Class Activation Maps (CAM)

- **O que é CAM?**: Uma técnica para gerar mapas de ativação que indicam as regiões discriminativas de uma imagem usadas pela CNN para identificar uma categoria específica.
- **Explicabilidade Visual**: O resultado é um mapa de calor (heatmap) que visualiza onde a CNN está "olhando" ao fazer uma previsão.
- O CAM permite a localização de objetos (desenhar uma caixa delimitadora) usando apenas **rótulos de nível de imagem** (classificação), sem a necessidade de anotações de caixas delimitadoras (bounding boxes) durante o treinamento.
- **Proposta Original**: O método foi introduzido por Zhou et al. em 2016.

Global Average Pooling (GAP)

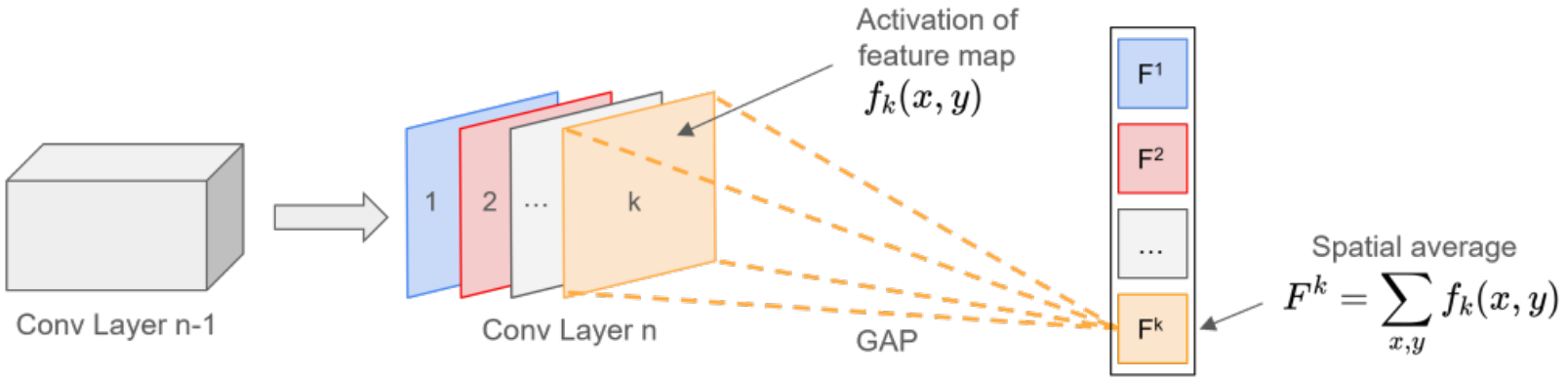
- **Restrição Arquitetural**: O método CAM original só pode ser aplicado em modelos com uma arquitetura específica.
- **Estrutura Obrigatória**: O modelo deve ter uma camada de *Global Average Pooling* (GAP) imediatamente após a última camada convolucional, seguida diretamente pela camada de classificação (Softmax ou FC).
- **Alternativa ao Achamento**: Essa estrutura evita as camadas FC densas que agem como caixas pretas entre o mapa de características e o *output*.



CAM via GAP. Fonte: Zhou et al. 2016.

Global Average Pooling (GAP): Funcionamento

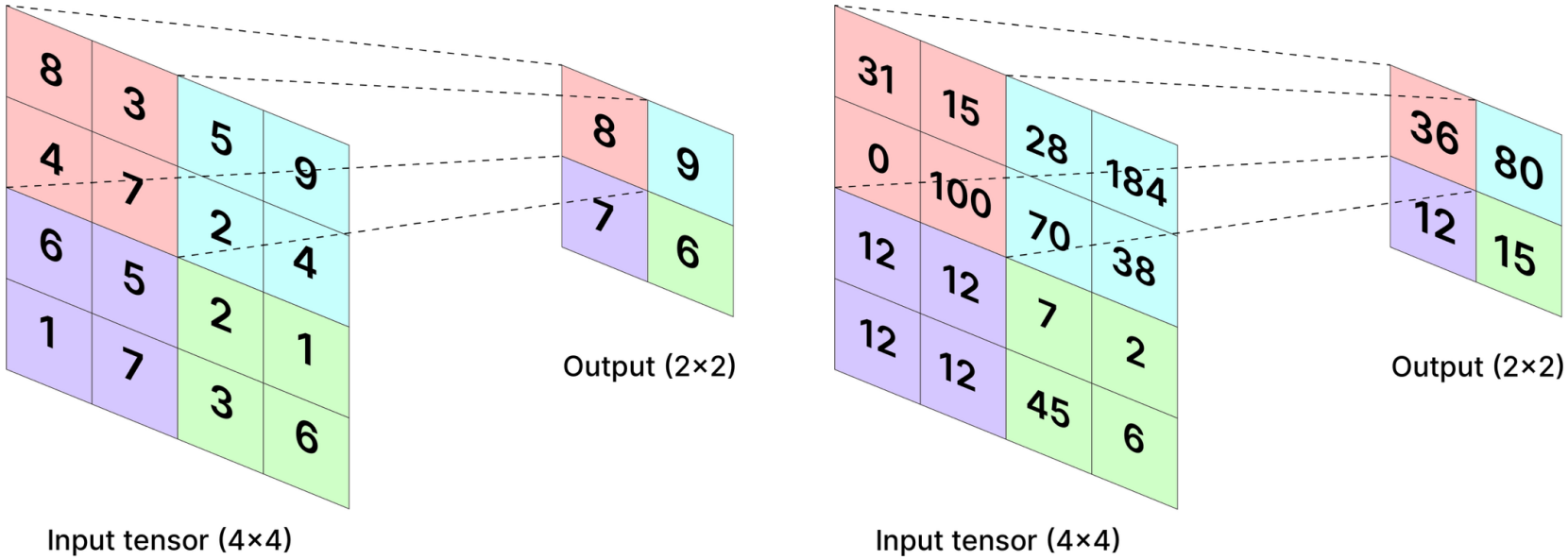
- **Operação GAP**: A camada GAP calcula a média espacial (média de todos os pixels) de cada mapa de características na última camada convolucional.
- **Redução Dimensional**: Transforma cada mapa de característica k de dimensão $H \times W$ em um único valor escalar F_k.



Operação GAP. Fonte: Joh Fischer.

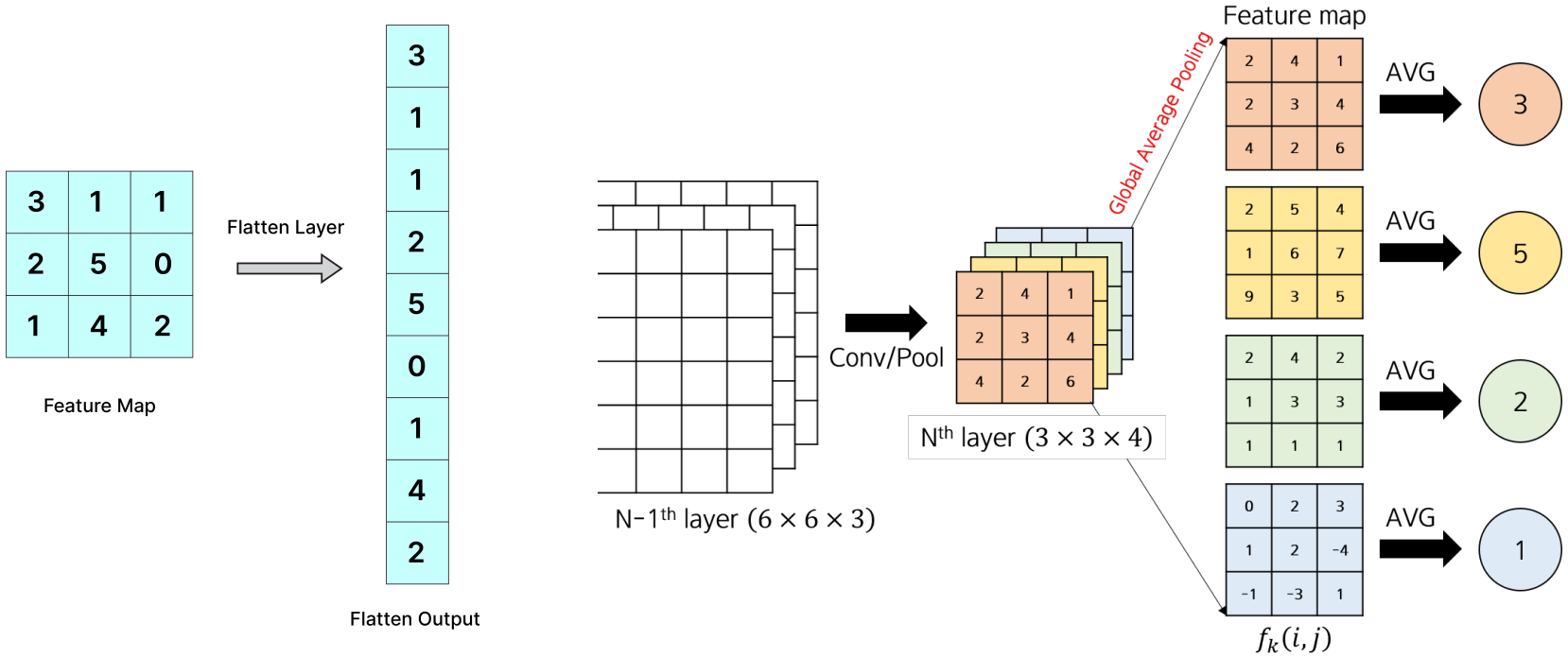
GAP versus Global Max Pooling (GMP)

- GAP tende a encorajar a rede a identificar a *extensão completa* de um objeto, pois a média se beneficia de todas as ativações positivas.
- GMP, por outro lado, pode se contentar em identificar apenas o ponto mais discriminativo.

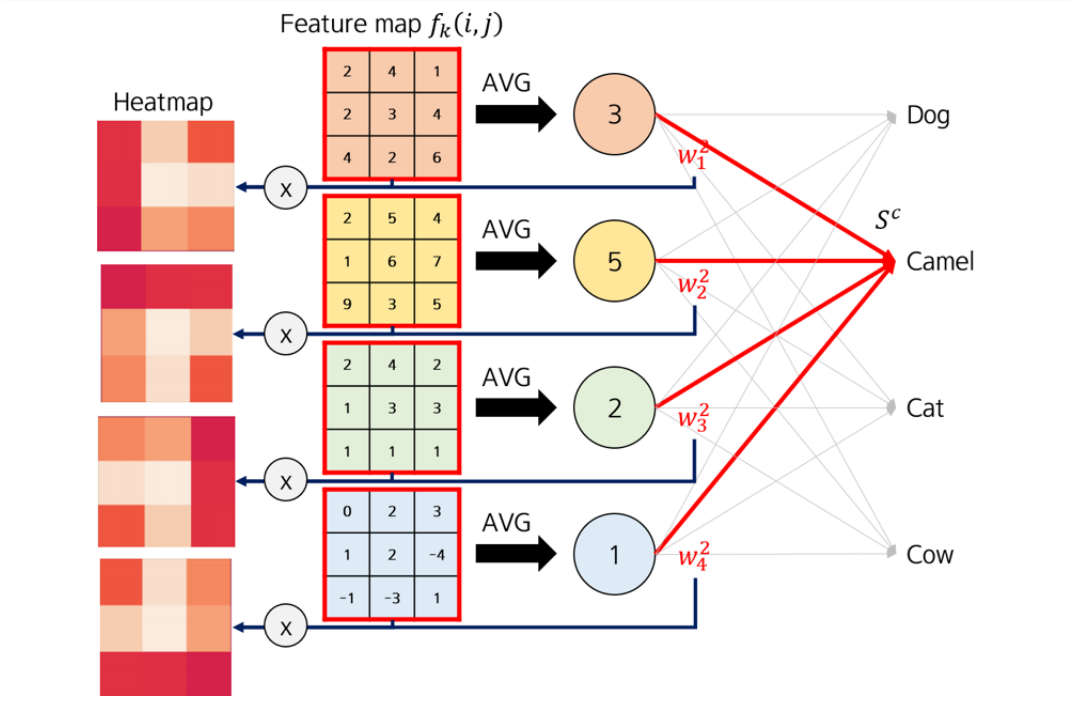


Comparação Flatten versus GAP

- Flatten perde informação espacial.
- GAP preserva a informação espacial. Para um tensor $(8, 10, 64)$, GAP produz um tensor $(1, 1, 64)$.

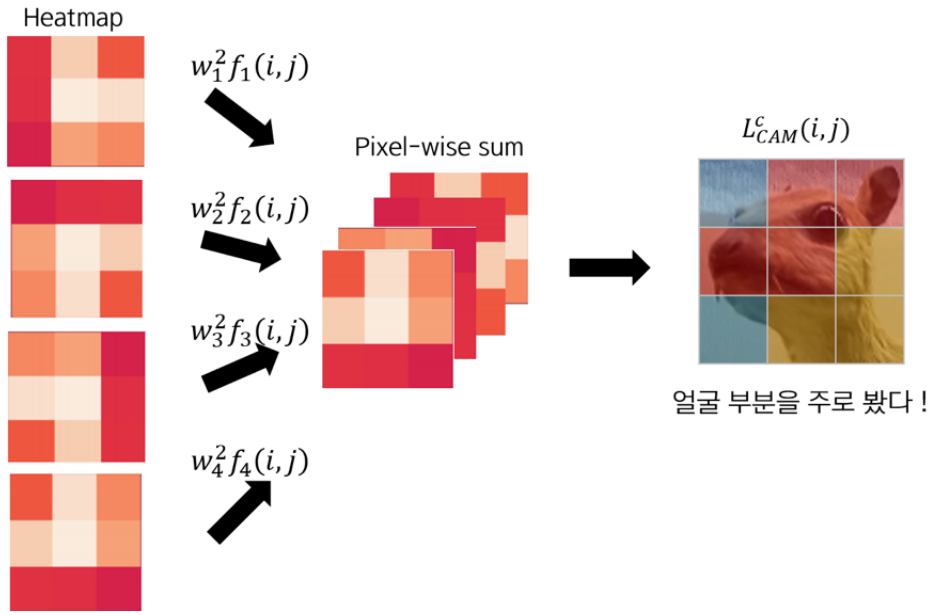


Global Average Pooling (GAP)



Operação GAP. Fonte: Taeyang Yang.

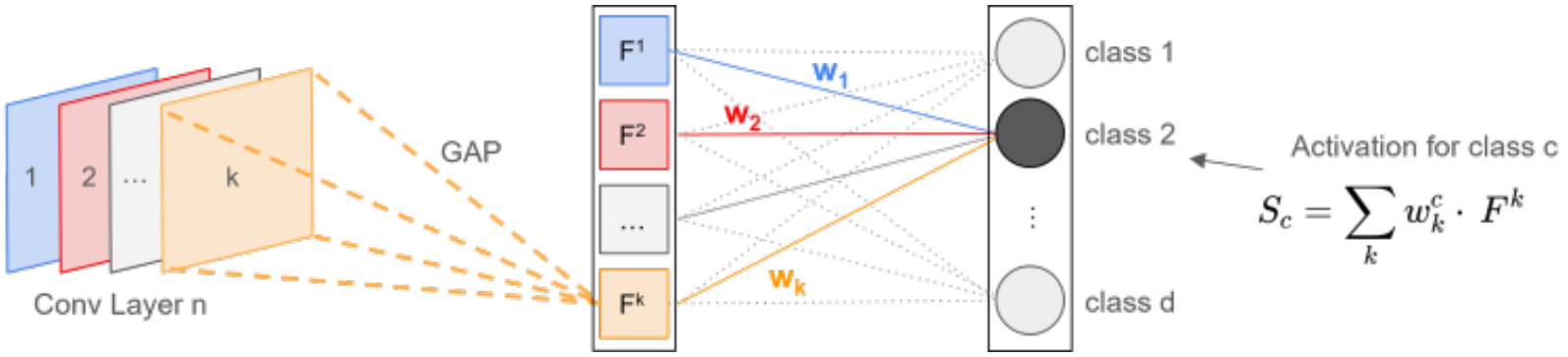
Global Average Pooling (GAP)



Exemplo de CAM. Fonte: Taeyang Yang.

Global Average Pooling (GAP): Classificação

- **Entrada para Classificação**: O vetor resultante dos valores F_k é então alimentado diretamente na camada de classificação (Softmax/FC).
- **Importância do Conceito**: O peso w_k^c representa a importância do mapa de características k (ou "conceito" k) para a previsão da classe c.



Operação GAP. Fonte: Joh Fischer.

A Equação do Class Activation Map

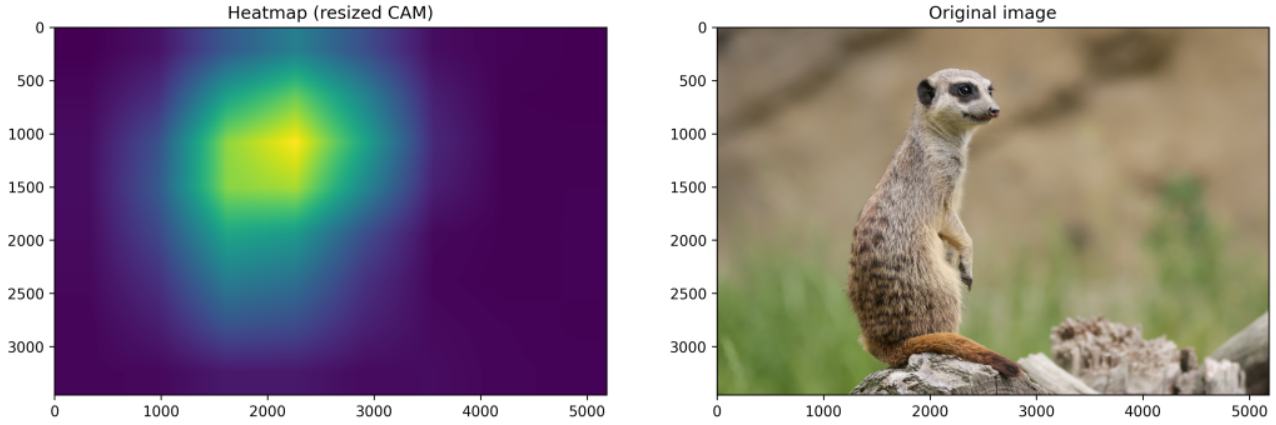
- O CAM Map M_c para a classe c em um local (x, y) é definido pela soma ponderada das ativações de todos os mapas de características:

$$M_c(x, y) = \sum_k w_k^c f_k(x, y)$$

- **Variáveis da Fórmula**:
  - $M_c(x, y)$: Class Activation Map, o valor de importância no local (x, y) para a classe c.
  - w_k^c: Peso da conexão da camada FC, representando a importância do k -ésimo mapa de características para a classe c.
  - $f_k(x, y)$: Ativação do k-ésimo mapa de características na última camada convolucional, na posição (x, y).
  - k: Índice sobre todos os mapas de características da última camada convolucional.
- **Importância do Conceito**: O peso w_k^c representa a importância do mapa de características k (ou "conceito" k) para a previsão da classe c.

CAM: Exemplo

- **Cálculo do CAM**: Realizar a soma ponderada das features pelo peso da classe.
- **Normalização e Redimensionamento**: O CAM resultante é de baixa resolução (e.g., 7x7) e deve ser normalizado (para mapeamento de cores) e redimensionado (upsampled) para as dimensões originais da imagem.
- **Geração do Mapa de Calor**: Usar uma paleta de cores (e.g., `jet`) para criar o mapa de calor.



Exemplo de CAM. Fonte: Joh Fischer.

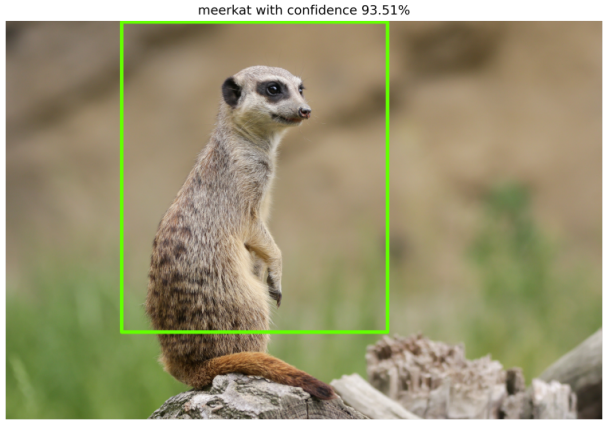
CAM: Exemplo

- **Sobreposição**: A sobreposição da imagem original com o mapa de calor visualiza as regiões discriminativas.
- Note que o CAM foca apenas nas ativações da última camada convolucional, que pode ser mais propensa às características de alto nível, mas ignora as representações localizadas de camadas mais rasas.

Exemplo de CAM. Fonte: Joh Fischer.

Weakly-supervised object localization (WSOL)

- **WSOL**: O CAM foi originalmente proposto como uma forma de realizar Localização de Objeto Supervisionada Fracamente (WSOL).
- **Fluxo de Trabalho de Localização**: Após gerar o mapa de calor, uma técnica de limiarização simples é usada: segmentar regiões com valor acima de 20% do valor máximo do CAM.
- **Geração da Bounding Box**: A caixa delimitadora é desenhada cobrindo o maior componente conectado na região segmentada, localizando o objeto discriminativo.



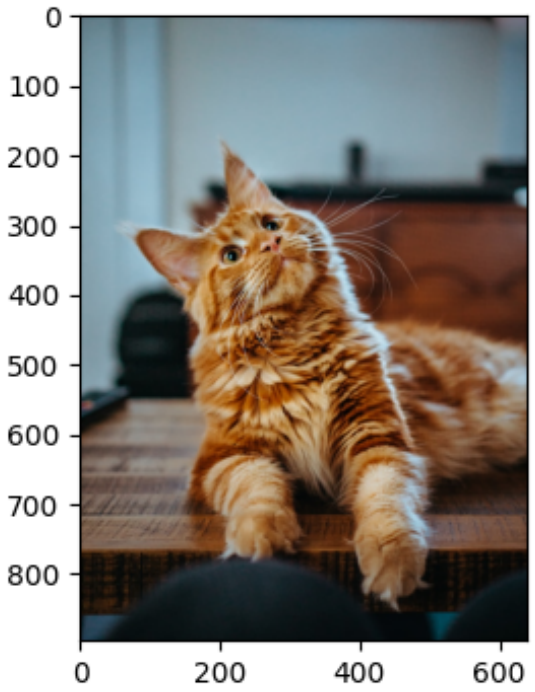
Bounding Box via CAM. Fonte: Joh Fischer.

CAM: Implementação Pytorch

Download da imagem de entrada aqui | Créditos: https://unsplash.com/@sadmax

In [1]:
```
import matplotlib.pyplot as plt
from PIL import Image

img = Image.open("amber-kipp-75715CVEJhI-unsplash.jpg")
plt.figure(figsize=(4,4))
plt.imshow(img)
plt.show()
```


0
100
200
300
400
500
600
700
800
0
200
400
600

In [2]:

```
import torch
import torch.nn.functional as F
import torchvision.models as models
import matplotlib.pyplot as plt
from torchvision import transforms
import numpy as np



# O ResNet é já contém a estrutura GAP antes de FC.
model = models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)
model.eval()

# Armazenar ativações da última camada convolucional ('layer4')
activation = {}
def get_activation(name):
    def hook(model, input, output):
        activation[name] = output.detach()
    return hook

# Registrar o hook na camada alvo (layer4)
model.layer4.register_forward_hook(get_activation('final_conv'))
```

Out[2]:

```
<torch.utils.hooks.RemovableHandle at 0x150ca9a50>
```

In [3]:

```
# Define transformações no padrão da ResNet (224x224)
preprocess = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225])
])

transformada_img = preprocess(img)

# Adiciona dimensão de batch
model_img = transformada_img.unsqueeze(0)
print(model_img.shape)
```

```
torch.Size([1, 3, 224, 224])
```

In [4]:

```python
# Faz o forward pass
outputs = model(model_img)
probs = F.softmax(outputs, dim=1)
class_idx = torch.argmax(probs).item()

# Gera a saída do modelo
output_shape = model(transformada_img.unsqueeze(0)).shape

print(f"Classe Prevista: {class_idx}")
```

```
Classe Prevista: 282
```

In [5]:

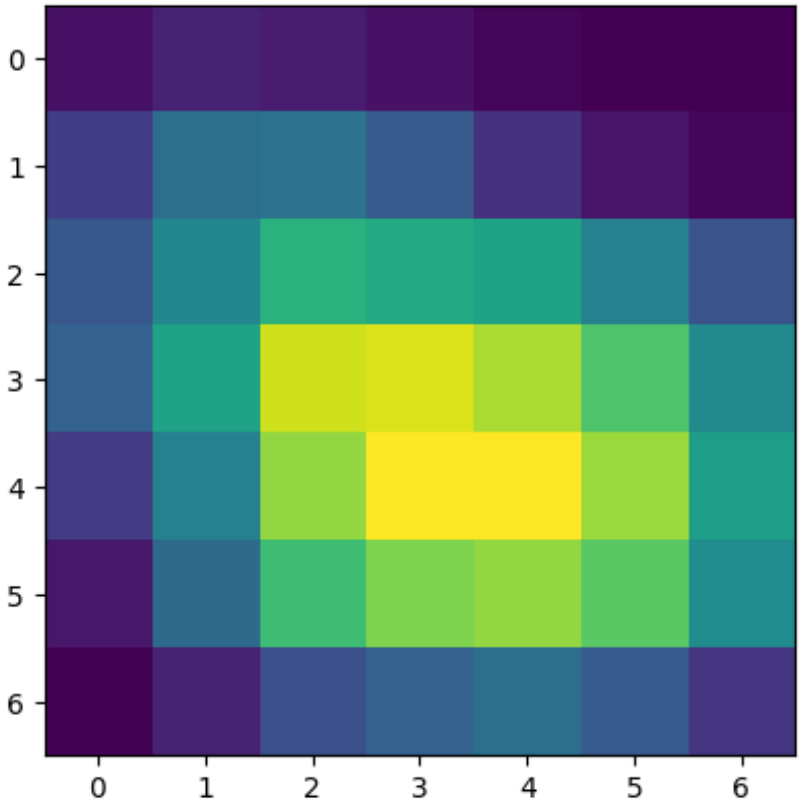
```python
# Extrai Feature Maps
conv_layer_output = activation["final_conv"]
conv_layer_output = conv_layer_output.squeeze(0)

# Recuperar Pesos da Camada FC
fc_weights = model.fc.weight
cat_class_idx = class_idx  # Índice da classe 'cat' na ResNet-18
# Isola os Pesos para a Classe Prevista
cat_fc_weights = fc_weights[cat_class_idx].unsqueeze(1).unsqueeze(1)

# Computa a soma ponderada para gerar o CAM
final_conv_layer_output = cat_fc_weights * conv_layer_output
class_activation_map = final_conv_layer_output.sum(0)
```

In [6]:

```
plt.imshow(class_activation_map.to("cpu").detach().numpy())
plt.show()
```



In [7]:

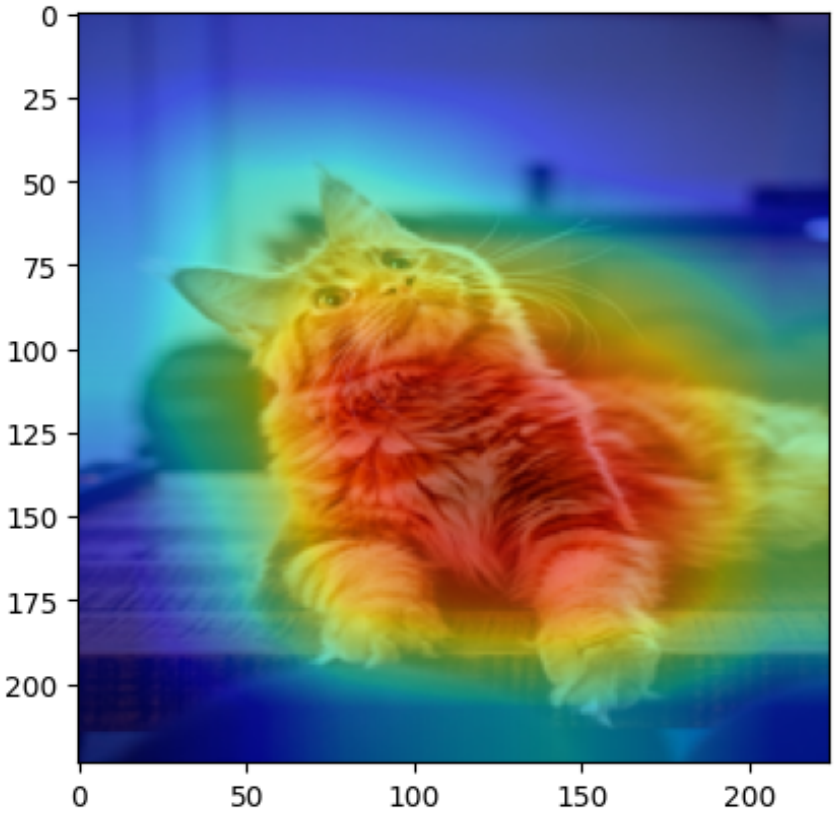
```
# Faz o resize do CAM para o tamanho da imagem de entrada
cam_resized = F.interpolate(
    class_activation_map.unsqueeze(0).unsqueeze(0),
      size=tuple(model_img.shape[-2:]),
      mode='bilinear',
      align_corners=False)

# Converte CAM para NumPy array
cam_np = cam_resized.squeeze().cpu().detach().numpy()
cam_expanded = np.expand_dims(cam_np, axis=2)

# Converte imagem de entrada para NumPy array
img_np = np.array(img.resize((224,224)))
```

In [8]:

```
# Mostra CAM + Imagem
plt.imshow(img_np)
plt.imshow(cam_expanded, alpha=0.5, cmap='jet')
plt.show()
```

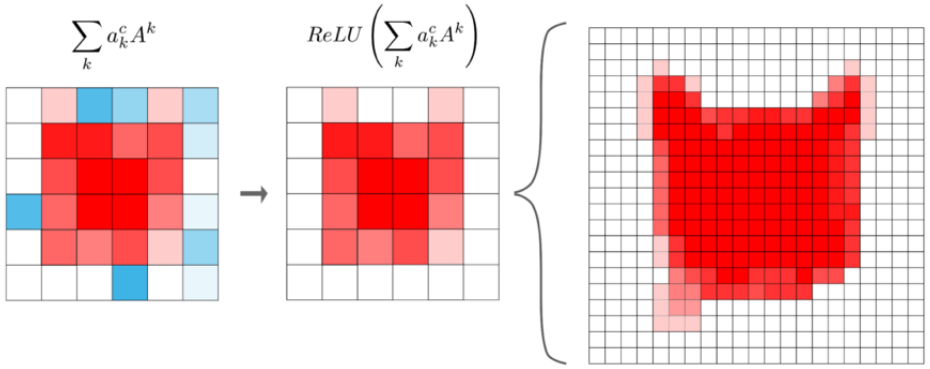


CAM: Limitações

- **Restrição de Arquitetura**: A principal limitação é a exigência de ter GAP seguido diretamente pelo Softmax, o que força modificações arquiteturais em modelos que não o possuem (e.g., AlexNet original ou VGG).
- **Necessidade de Re-treinamento/Ajuste Fino**: Se um modelo popular não se adequar à arquitetura CAM, ele deve ser modificado (removendo camadas FC) e então ajustado (*fine-tuning*).
- **Baixa Resolução**: O mapa de calor é inicialmente gerado na baixa resolução dos feature maps da última convolução (e.g., 7x7 ou 14x14).
- **Dependência da Última Camada**: O CAM foca apenas nas ativações da última camada convolucional, que pode ser mais propensa a características de alto nível, mas ignora as representações localizadas de camadas mais rasas.

Evolução e Outras Abordagens: Grad-CAM

- **Grad-CAM (Generalização)**: O Grad-CAM (Gradient-weighted Class Activation Mapping) foi proposto para superar essa restrição, tornando a localização visual aplicável a *qualquer* arquitetura de CNN.
- Em vez de usar os pesos w_k^c da camada FC (como no CAM), o Grad-CAM usa os gradientes da pontuação da classe alvo em relação aos feature maps da última convolução para calcular os coeficientes de importância α_k^c.

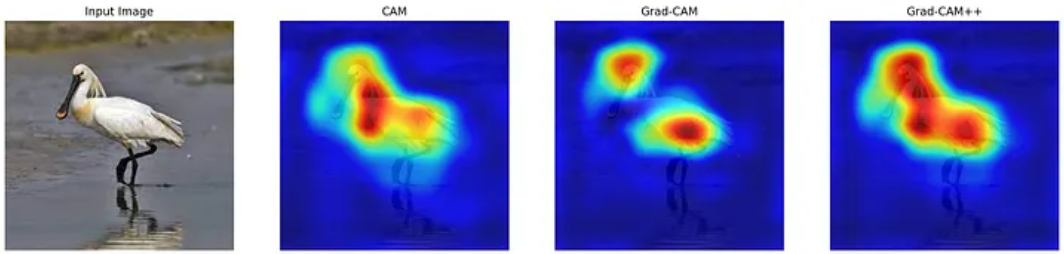


Exemplo de Grad-CAM. Fonte: A Data Odyssey.

Evolução e Outras Abordagens: Grad-CAM++

Grad-CAM++: Uma melhoria do Grad-CAM que calcula uma média ponderada *verdadeira* dos gradientes, oferecendo melhores explicações visuais, especialmente em imagens com múltiplas ocorrências do mesmo objeto.

- **Métodos Sem Gradiente (Gradient-Free)**: Surgiram métodos que eliminam a dependência de gradientes para evitar problemas como saturação de gradiente.
- **Score-CAM**: Utiliza o score de forward pass de cada mapa de característica para determinar sua importância, superando as questões de gradiente.
- **Recipro-CAM**: Um método mais recente, considerado *state-of-the-art* em eficiência computacional e precisão em certas métricas (ADCC).



Exemplo de Grad-CAM. Fonte: Tanishq Sardana.

Conclusão

- **Fundamento de XAI**: CAM é um método pioneiro e eficaz para visualizar as decisões de classificação em CNNs.
- **Interpretabilidade por Design**: Explora uma arquitetura de rede específica (GAP) que mantém a capacidade de localização.
- **Localização Poderosa**: Permite a localização de objetos (WSOL) com alta precisão, utilizando apenas rótulos de nível de imagem.
- **Insight em Pesos**: Liga diretamente a importância do conceito (pesos FC) com a presença espacial (feature maps).
- **Evolução**: Embora limitado arquiteturalmente, ele pavimentou o caminho para métodos mais flexíveis e generalizados como Grad-CAM.

Material Adicional

- Class Activation Mapping (CAM): Better Interpretability in Deep Learning Models
- Implementation of Class Activation Map (CAM) with PyTorch
- Building Blocks of Interpretability
- Understanding Class Activation Maps (CAMs) for Deep Learning Interpretability

Tarefa 1: Geração e Visualização do CAM

Gere CAM em um modelo popular (exemplo, VGG16) para localizar a região discriminativa que justifica a previsão de uma classe ImageNet.

Tarefa 2: Comparação CAM vs. Grad-CAM

Compare visualmente e analiticamente a localização fornecida pelo CAM (restrito à arquitetura GAP) e pelo Grad-CAM (abordagem mais generalizada) em diferentes cenários.