CS6450- Visual Computing-Presentation 2

LayerCAM : Exploring Hierarchical Class Activation Maps for Localization

Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei

IEEE Transactions on Image Processing (Volume: 30),June 2021

Presented on April 21,2022

Guide: Prof.C.Krishna Mohan, Dept.of CSE, IITH

Teaching Assistant: Mr.Udaya Kumar Ambati

Presented by: Prasanna Kumar R [SM21MTECH14001]

Table of Contents

Motivation

- Current deep models are very successful in Computer Vision, NLP etc.
- One problem with these networks is the explainability of the results it produces.
- We need explainable or interpretable deep models to gain human trust.
- One of the works which could explain these deep models is the LayerCAM

Previous Works: CAM, GradCAM and GradCAM++

- **Class Activation Mapping (CAM)[1]** localizes the discriminative regions used by a particular task despite not being trained for them.
- **Gradient -weighted Class Activation Mapping (GradCAM)[2]** uses gradient of target concept flowing into final convolutional layer.
- GradCAM is a generalization of CAM for any CNN architecture.
- **GradCAM++ [3]** is same as GradCAM except for different weightage for each pixel in the feature map

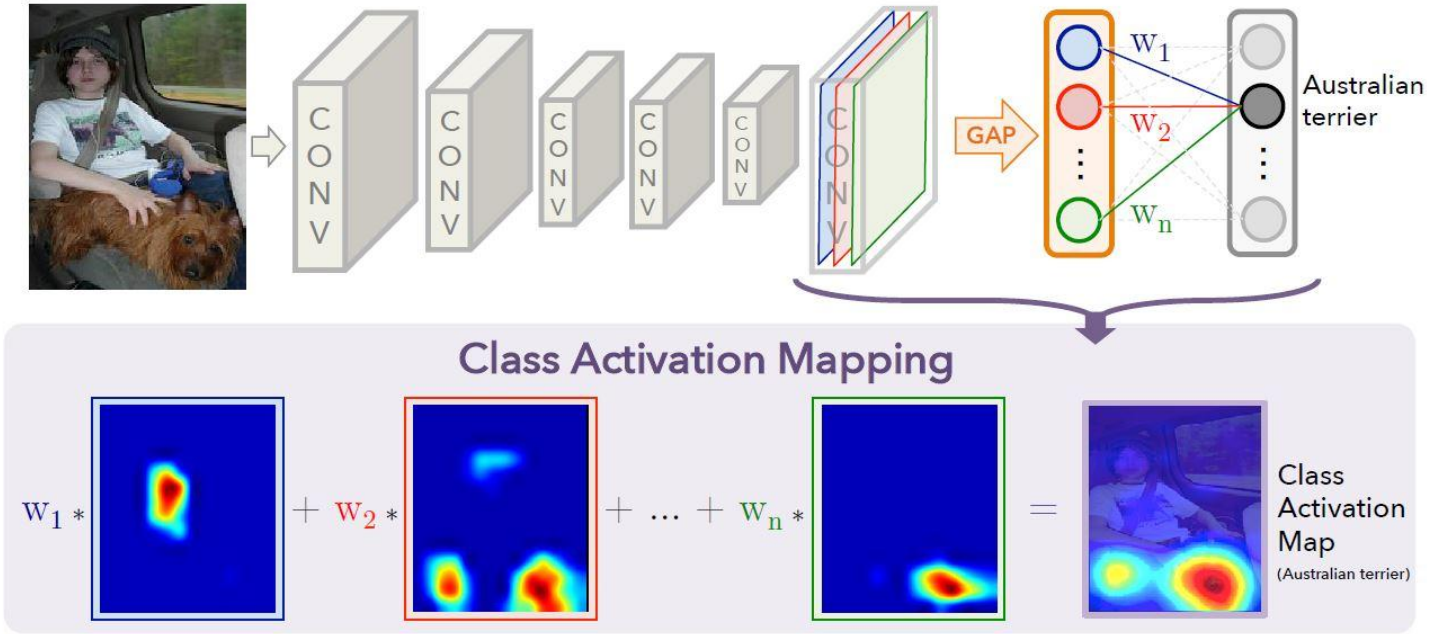
[1]*B. Zhou et al, "Learning deep features for discriminative localization," CVPR 2016*

[2]*R. R. Selvaraju, et al "Grad-cam: Visual explanations from deep networks via gradient-based localization," ICCV 2017*

[3]*A. Chattopadhay et al "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," WACV,2018*

CAM

Helps us to know what the network is looking at while predicting the class



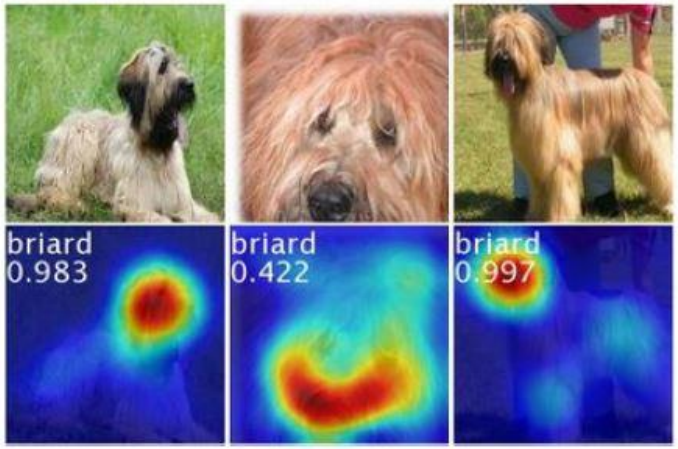
Picture Credits: *B. Zhou et al, "Learning deep features for discriminative localization," CVPR 2016*

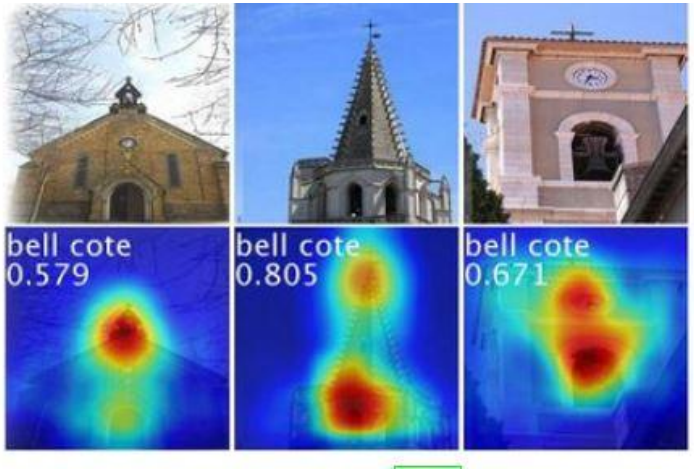
CAM:

Mathematical Expression:

$$Y^c = \sum_k w_k^c \cdot \frac{1}{Z} \sum_i \sum_j A_{ij}^k$$

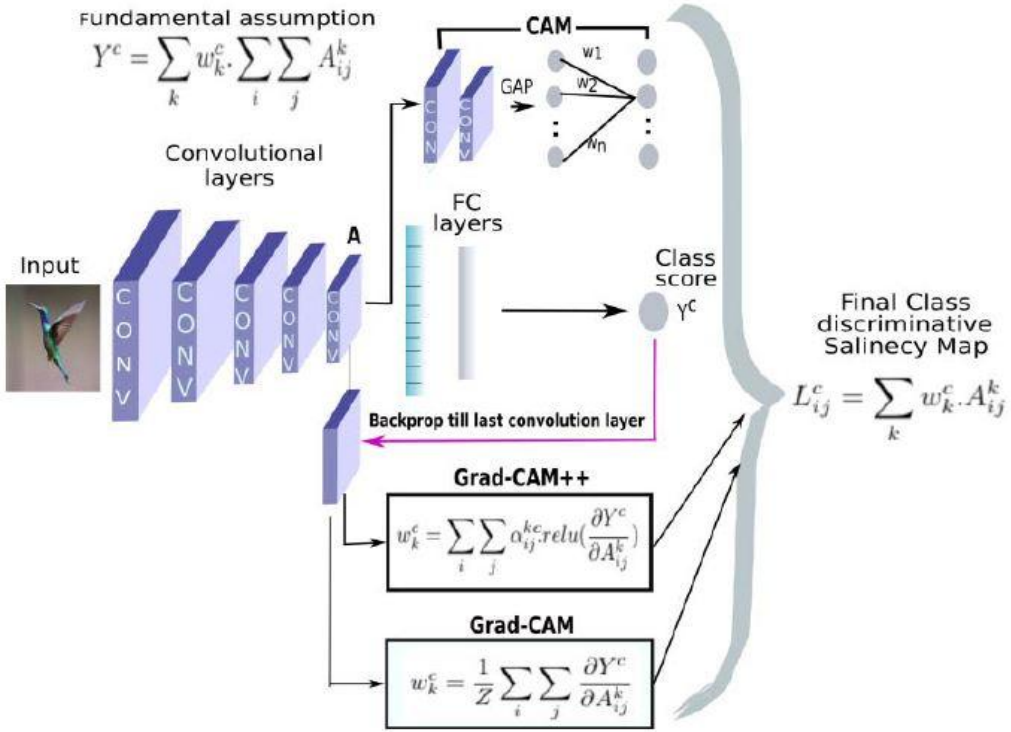
Examples:





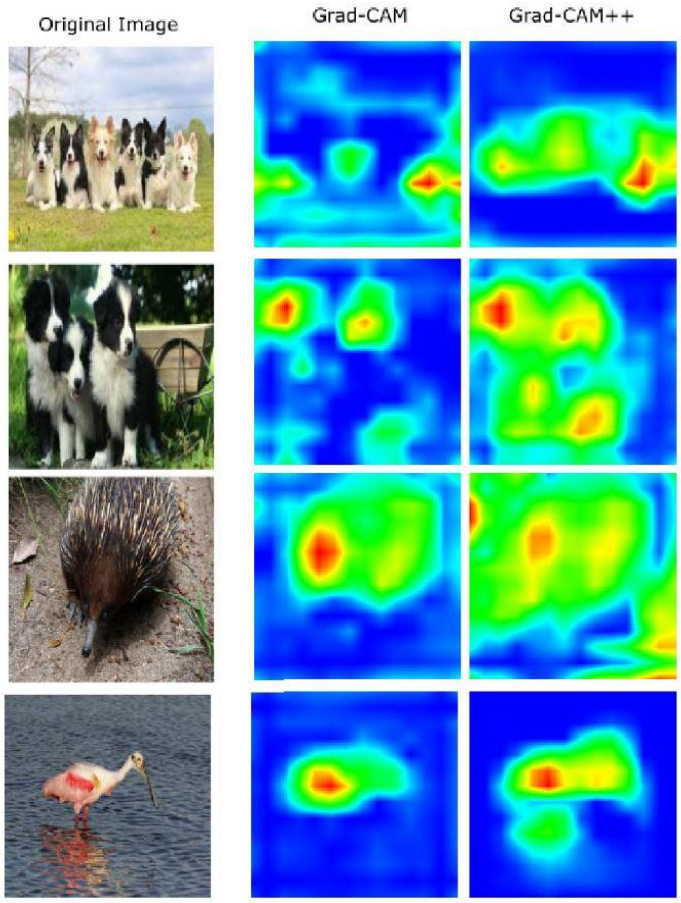
Picture Credits: *B. Zhou et al, "Learning deep features for discriminative localization," CVPR 2016*

Overview of discussed methods:



Picture Credits: *A. Chattopadhay et al "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," WACV,2018*

Performance of GradCAM and GradCAM++



LayerCAM:

- It rethinks the relationship between feature map and their corresponding gradients.
- It can produce reliable class activation maps for different layers of CNN instead of only the final convolutional layer.
- This property enables us to collect object localization information from coarse (rough spatial localization) to fine (precise fine-grained details) levels.

Credits: *P. -T. Jiang et al, "LayerCAM: Exploring Hierarchical Class Activation Maps for Localization," in IEEE Transactions on Image Processing, vol. 30, 2021.*

LayerCAM:

- Instead of a global weight, we generate a separate weight for each spatial location in a feature map.
- Formally, the weight of the spatial location (i, j) in the k-th feature map can be written as

$$w_{ij}^{kc} = relu(g_{ij}^{kc}). \quad \text{where,} \quad g_{ij}^{kc} = \frac{\partial y^c}{\partial A_{ij}^k}$$

- To obtain the class activation map for a certain layer, Layer-CAM first multiplies the activation value of each location in the feature map by a weight:

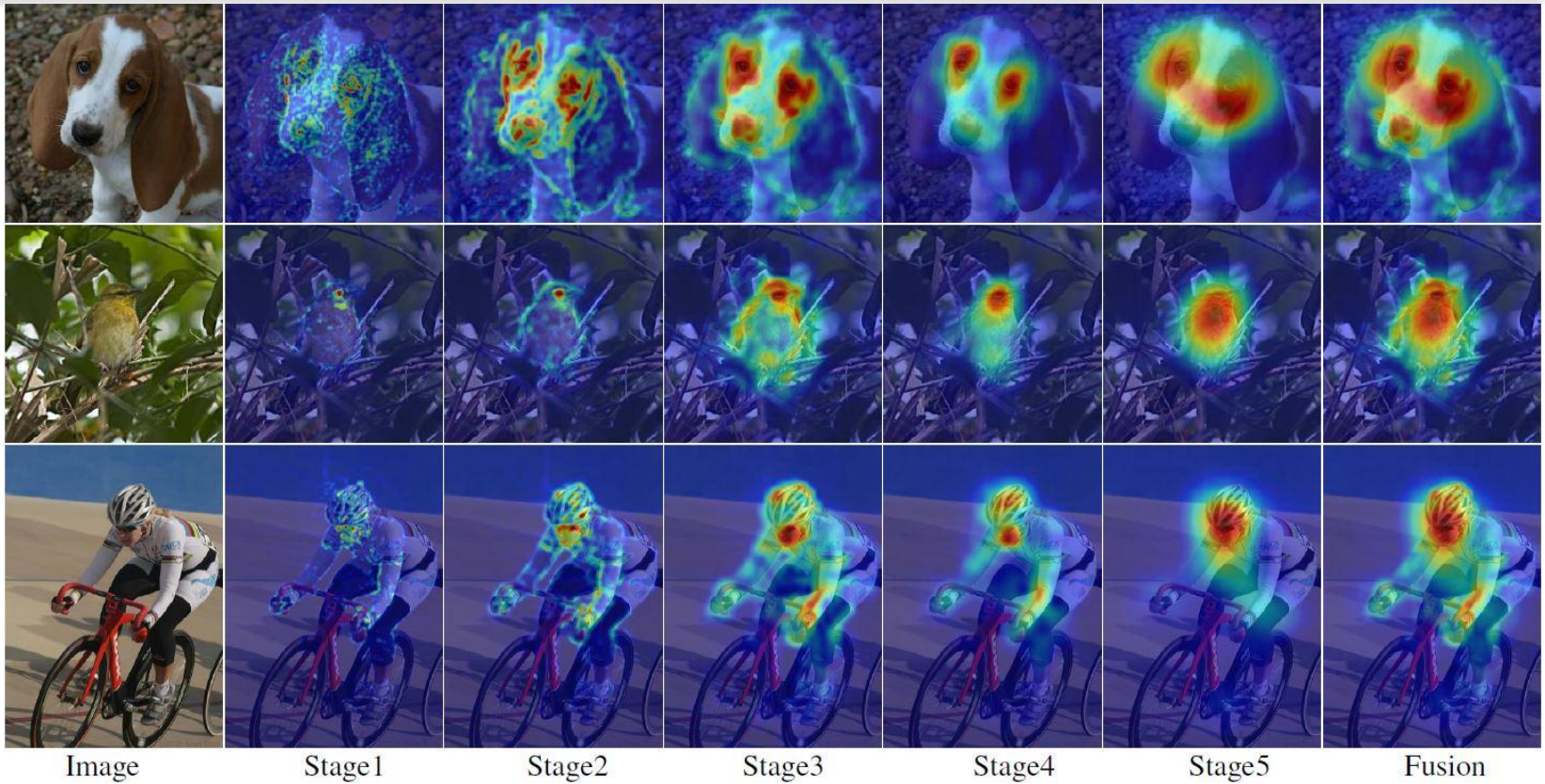
$$\hat{A}_{ij}^k = w_{ij}^{kc} \cdot A_{ij}^k.$$

LayerCAM:

- Finally, the results $\hat{A_k}$ are linearly combined along the channel dimension to obtain the class activation map, which is formulated as follows:

$$M^c = \text{ReLU}\left(\sum_k \hat{A}^k\right)$$

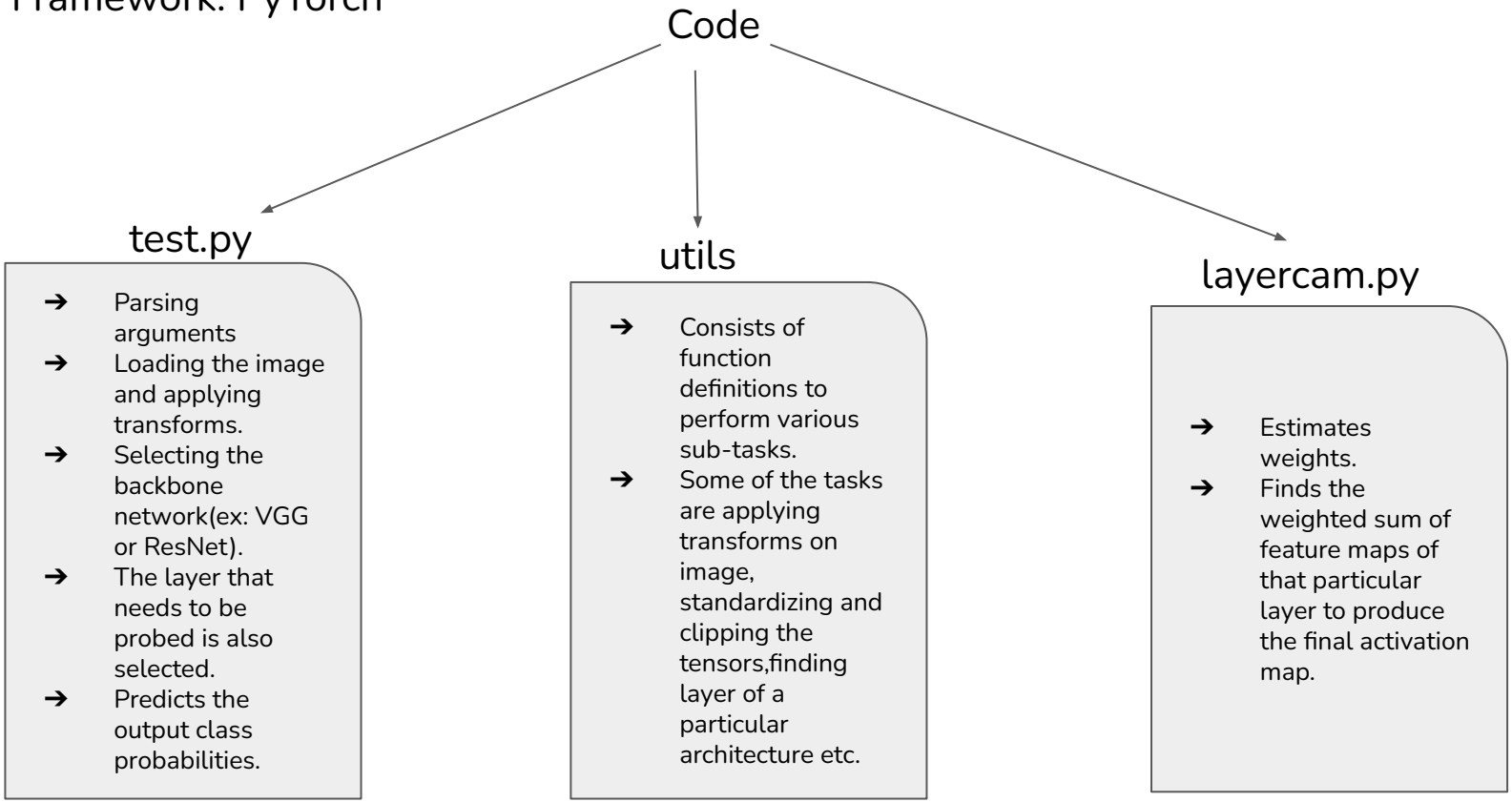
LayerCAM:



Credits: *P. -T. Jiang et al, "LayerCAM: Exploring Hierarchical Class Activation Maps for Localization," in IEEE Transactions on Image Processing, vol. 30, 2021.*

Implementation:

- The code consists of 3 main files.
- Framework: PyTorch



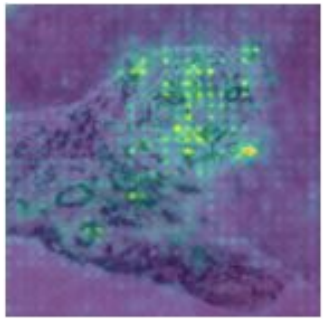
LayerCAM visualization on test images:

Input Image

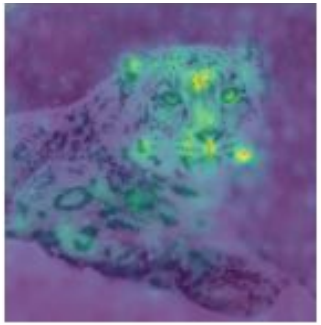
```
!python test.py --img_path='/content/drive/MyDrive/LayerCAM/snow_leopard.jpg'

0.975 -> snow leopard
0.020 -> leopard
0.002 -> lynx
0.002 -> cheetah
0.000 -> jaguar
```

Class probabilities

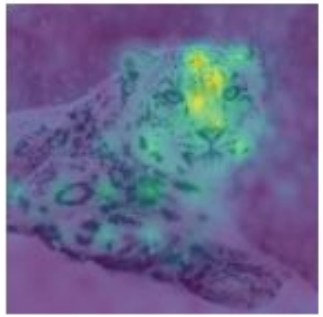
Obtained Results:



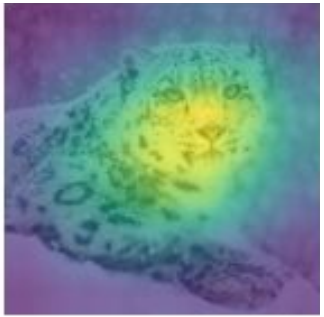
Layer 1



Layer 2

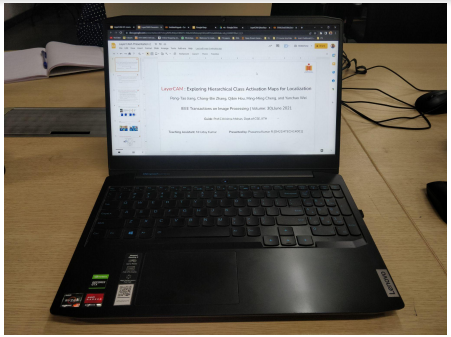


Layer 3



Layer 4

LayerCAM visualization on test images:

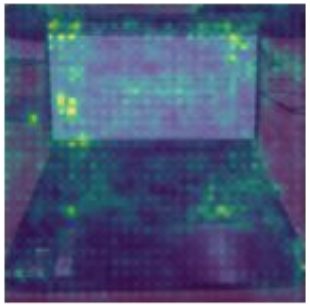


Input Image

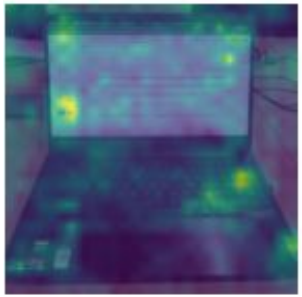
```
!python test.py --img_path='/content/drive/MyDrive/LayerCAM/laptop.jpg'

0.513 -> laptop computer
0.432 -> notebook computer
0.021 -> computer mouse
0.011 -> desktop computer
0.008 -> monitor
```

Class probabilities

Obtained Results:



Layer 1



Layer 2

Layer 3

Layer 4

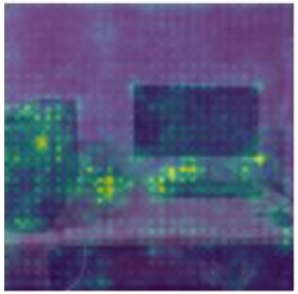
LayerCAM visualization on test images:

Input Image

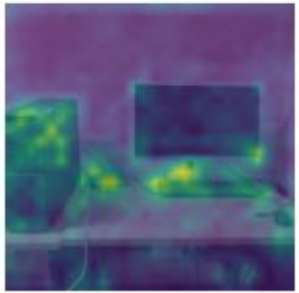
```
!python test.py --img_path='/content/drive/MyDrive/LayerCAM/desktop.jpg'

0.747 -> desktop computer
0.057 -> speaker
0.043 -> CRT screen
0.037 -> desk
0.036 -> monitor
```

Class probabilities

Obtained Results:



Layer 1



Layer 2

Layer 3

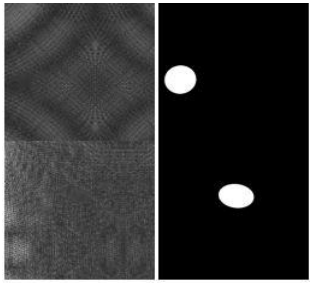
Layer 4

Experimental Analysis:

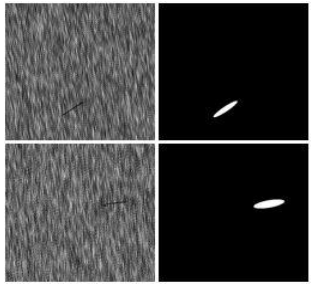
Industry Surface Defect Localization Experiment:

Dataset: DAGM 2007 defect dataset: 3550 Training images and 400 testing images

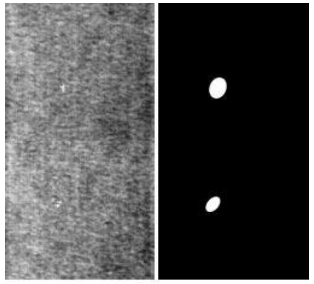
No. of problems in the dataset: 6



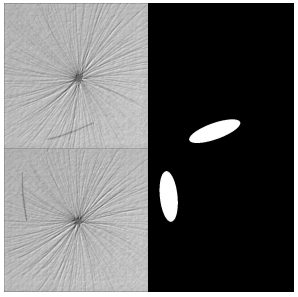
P1



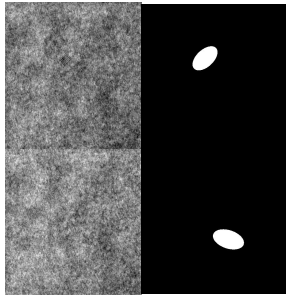
P2



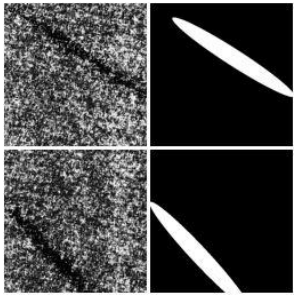
P3



P4



P5



P6

Industry Surface Defect Localization Experiment: Model Training

Dataset: DAGM 2007 defect dataset

Backbone: ResNet-50

Batch size: 8

Epochs: 10

Optimizer: SGD

Industry Surface Defect Localization Experiment: Model Training

```
!python main.py
[Starting Problem 1 ...]
[Creating the model ...]
resnet50
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth
100% 97.8M/97.8M [00:00<00:00, 136MB/s]
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will create 4 worker processes in total. Our s
  cpuset_checked))
### Start Training [...] ###
Epoch 0 - Train Accuracy: 88.32298136645963,    Loss: 0.33976896285825636
Epoch 0 - Val Accuracy: 98.25581395348837,    Loss: 0.13242504000663757
Epoch 1 - Train Accuracy: 98.63354037267081,    Loss: 0.06288437811160309
Epoch 1 - Val Accuracy: 99.4186046511628,    Loss: 0.0318148136138916
Epoch 2 - Train Accuracy: 99.5031055900621,    Loss: 0.021492081553738863
Epoch 2 - Val Accuracy: 100.0,    Loss: 0.030383452773094177
Epoch 3 - Train Accuracy: 100.0,    Loss: 0.006889878431774722
Epoch 3 - Val Accuracy: 100.0,    Loss: 0.013441832736134529
Epoch 4 - Train Accuracy: 99.62732919254658,    Loss: 0.00943128498595046
Epoch 4 - Val Accuracy: 98.83720930232558,    Loss: 0.06346524506807327
Epoch 5 - Train Accuracy: 100.0,    Loss: 0.0062027313679988896
Epoch 5 - Val Accuracy: 100.0,    Loss: 0.022778458893299103
Epoch 6 - Train Accuracy: 100.0,    Loss: 0.003665190365929041
Epoch 6 - Val Accuracy: 100.0,    Loss: 0.006951421964913607
Epoch 7 - Train Accuracy: 100.0,    Loss: 0.002852469099026104
Epoch 7 - Val Accuracy: 100.0,    Loss: 0.006743794772773981
Epoch 8 - Train Accuracy: 100.0,    Loss: 0.0028814652463224548
Epoch 8 - Val Accuracy: 100.0,    Loss: 0.003244054736569524
Epoch 9 - Train Accuracy: 100.0,    Loss: 0.0028103066573602073
Epoch 9 - Val Accuracy: 100.0,    Loss: 0.010304621420800686
```

Industry Surface Defect Localization Experiment: Model Training

```
!python main.py
### Test metrics ###
Best Val Accuracy:100.0
Accuracy: 100.0
AUC-ROC: 1.0
Confusion Matrix:
[[147   0]
 [  0  26]]
Recall: 1.0
Precision: 1.0
### Time Metrics ###
Time to train: 1317.1238894462585
Inference Time Mean: 0.5620936399156397, STD:0.24383179598860039
[Starting Problem 2 ...]
[Creating the model ...]
resnet50
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will create 4 worker processes in total. Our sugg
  cpuset_checked))
### Start Training [...] ###
Epoch 0 - Train Accuracy: 87.20496894409938,    Loss: 0.4258043404228939
Epoch 0 - Val Accuracy: 87.79069767441861,    Loss: 0.7479735612869263
Epoch 1 - Train Accuracy: 89.8136645962733,    Loss: 0.29432888547456043
Epoch 1 - Val Accuracy: 98.83720930232558,    Loss: 0.06822416931390762
Epoch 2 - Train Accuracy: 97.88819875776397,    Loss: 0.06902266920006238
Epoch 2 - Val Accuracy: 100.0,    Loss: 0.002635089447721839
Epoch 3 - Train Accuracy: 99.87577639751552,    Loss: 0.018690122633894777
Epoch 3 - Val Accuracy: 100.0,    Loss: 0.002312957774847746
Epoch 4 - Train Accuracy: 99.5031055900621,    Loss: 0.01978427083376124
Epoch 4 - Val Accuracy: 100.0,    Loss: 0.001430334523320198
Epoch 5 - Train Accuracy: 99.37888198757764,    Loss: 0.017374509479344376
```

Test metrics for each model built:

```
### Test metrics ###
Best Val Accuracy:100.0
Accuracy: 100.0
AUC-ROC: 1.0
Confusion Matrix:
[[147   0]
 [  0  26]]
Recall: 1.0
Precision: 1.0
```

P1,P2,P3 and P4

```
### Test metrics ###
Best Val Accuracy:95.35
Accuracy: 93.64
AUC-ROC: 0.7884615384615384
Confusion Matrix:
[[147   0]
 [ 11  15]]
Recall: 1.0
Precision: 0.930379746835443
```

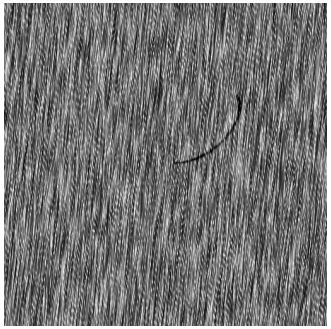
P5

```
### Test metrics ###
Best Val Accuracy:100.0
Accuracy: 98.27
AUC-ROC: 0.9423076923076923
Confusion Matrix:
[[147   0]
 [  3  23]]
Recall: 1.0
Precision: 0.98
```

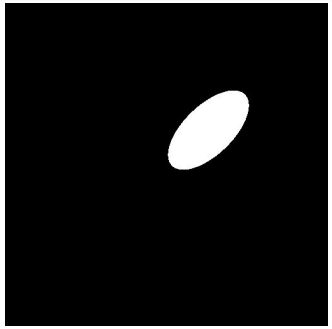
P6

Therefore, six trained models are obtained and LayerCAM is visualized in these models.

LayerCAM visualization on the built models

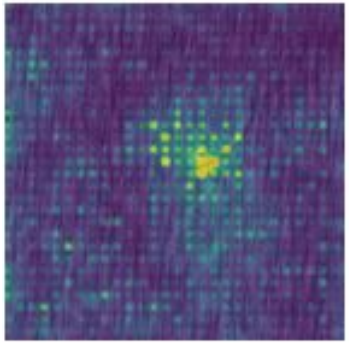


Input Image-P2

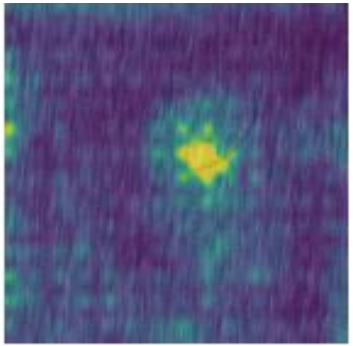


Ground Truth

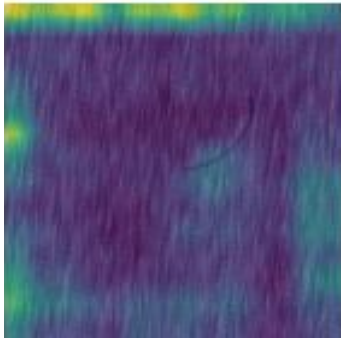
Obtained Results:



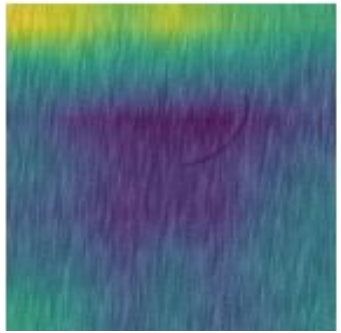
Layer 1



Layer 2

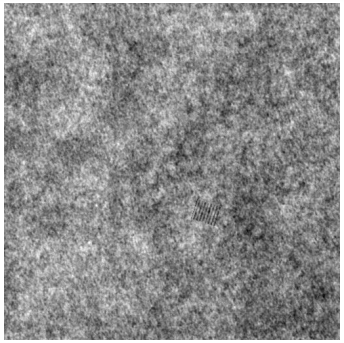


Layer 3

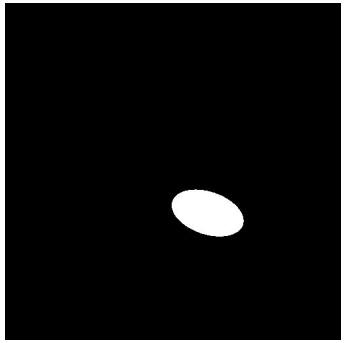


Layer 4

LayerCAM visualization on the built models

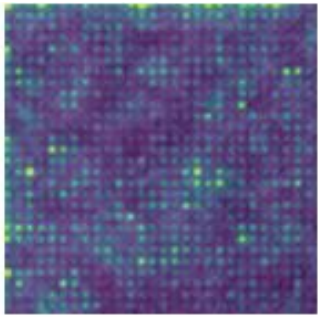


Input Image

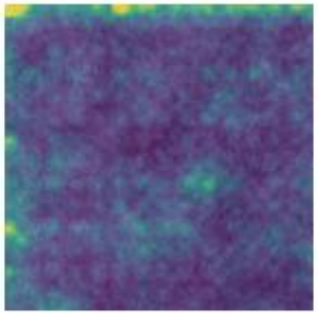


Ground Truth

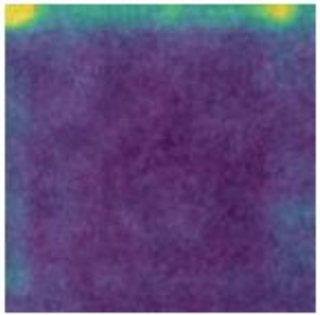
Obtained Results:



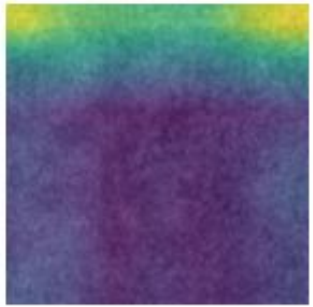
Layer 1



Layer 2



Layer 3



Layer 4

Experimental Results:

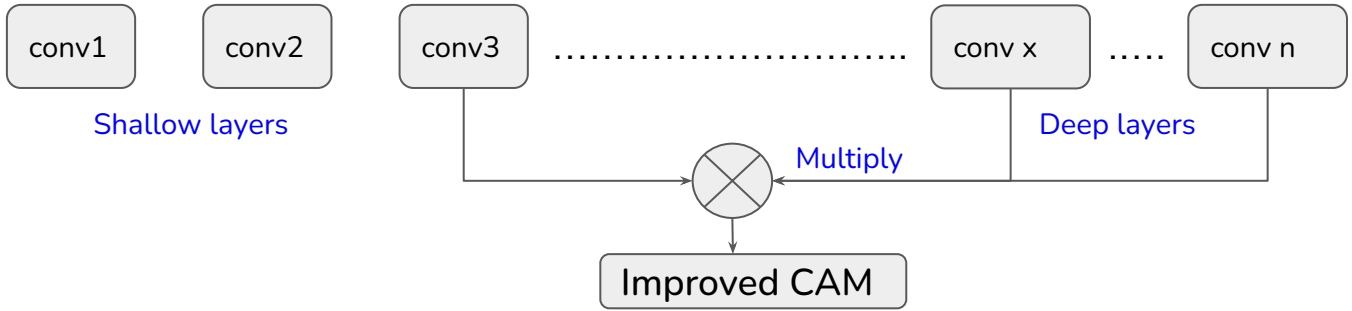
The generated maps are first thresholded to a binary mask and then we compute the IoU score among the binary mask and the ground truth mask.

Methods	mIoU (%)
SegNet	21.95*
RefineNet	**32.90***
LayerCAM	**27.26**

- The experimental results demonstrate that LayerCAM can locate defects more accurately than Grad-CAM and Grad-CAM++.
- Compared with fully supervised methods such as SegNet[4] and RefineNet[7], our LayerCAM achieves comparable performance faster.

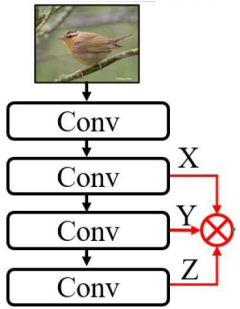
Proposed Novelty:

- Though LayerCAM captures the fine grained information from the shallow layers, with respect to the WSOL, it pays only little attention to the shallow features.
- Little attention because the shallow features are buried in the considerable background noise through conventional fusion strategy.
- I propose a multiplicative feature fusion instead of addition or concatenation to explicitly embed shallow features into deep ones.
- The multiplicative feature fusion strategy can take great advantage of shallow features since it treats the shallow and deep features in a synergistic way.



Proposed Novelty:

- For simplicity, let us consider the configuration as below:



- Features of different branches (*i.e.* X, Y, Z) are firstly unsampled to the same resolution (H,W) then combined by element-wise multiplication.

$$F_{mul} = \frac{1}{H \times W} \sum_{i,j}^{H,W} (X_{ij} \cdot Y_{ij} \cdot Z_{ij})$$

$$F_{add} = \frac{1}{H \times W} \sum_{i,j}^{H,W} (X_{ij} + Y_{ij} + Z_{ij})$$

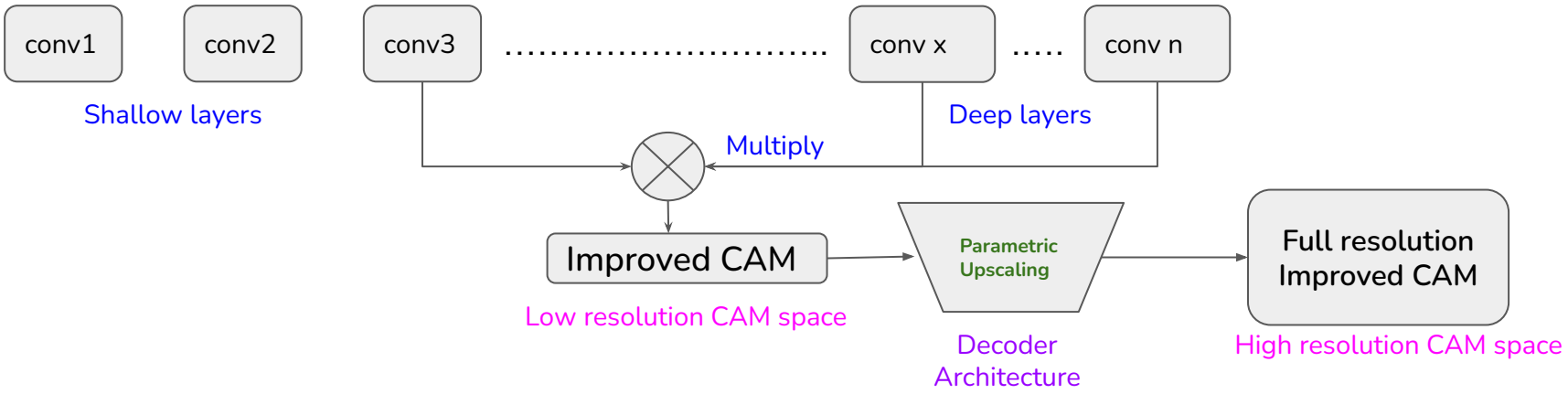
$$\frac{\partial F_{mul}}{\partial X_{ij}} = \frac{1}{H \times W} Y_{ij} \cdot Z_{ij}$$

$$\frac{\partial F_{add}}{\partial X_{ij}} = \frac{1}{H \times W}$$

different branches/layers are strongly coupled through the multiplicative operation

Proposed Novelty:

- Another Problem: Due to convolution and pooling operations, the backbones yield low resolution CAMs contributing to inaccurate localizations on WSOL tasks.
- S. Belharbi *et al* [8] introduced a generic method for parametric upscaling of CAMs that allows constructing accurate full resolution CAMs (FCAMs).
- In particular, they proposed a trainable decoding architecture that can be connected to any CNN classifier to produce highly accurate CAM localizations
- I propose that improved CAMs as a result of multiplicative fusion can be parametrically upscaled to achieve a Full Resolution Improved CAM.



LayerCAM: Conclusion

- In this paper,an attention method is proposed named LayerCAM, which can generate reliable class activation maps from different layers of the CNN effectively.
- The class activation maps from deep layers can locate the general location of objects, and the maps from shallow layers can generate fine-grained object localization information.
- The combination of class activation maps from different layers can find more object locations, which is beneficial for improving the performance of the weakly-supervised tasks.
- Experiments show that LayerCAM has better object localization ability than current attention methods.

References:

1. *B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in CVPR, 2016*
2. *R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” ICCV,2017*
3. *A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, “Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks,” in WACV,2018*
4. *V. Badrinarayanan et al, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, 2017.*
5. *H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, “Score-cam: Score-weighted visual explanations for convolutional neural networks,” in CVPR,2020*
6. *Wonho Bae, Junhyug Noh, and Gunhee Kim. Rethinking class activation mapping for weakly supervised object localization. ECCV, 2020.*
7. *G. Lin, A. Milan, C. Shen, and I. Reid, “Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation,” in CVPR, 2017.*
8. *S. Belharbi, A. Sarraf, M. Pedersoli, I. Ben Ayed, L. McCaffrey and E. Granger, "F-CAM: Full Resolution Class Activation Maps via Guided Parametric Upscaling," 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022.*

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Thank You!