Deep Learning for medical image recognition

Chen DANG & Hippolyte DEBERNARDI

Supervised by Ronan SICRE

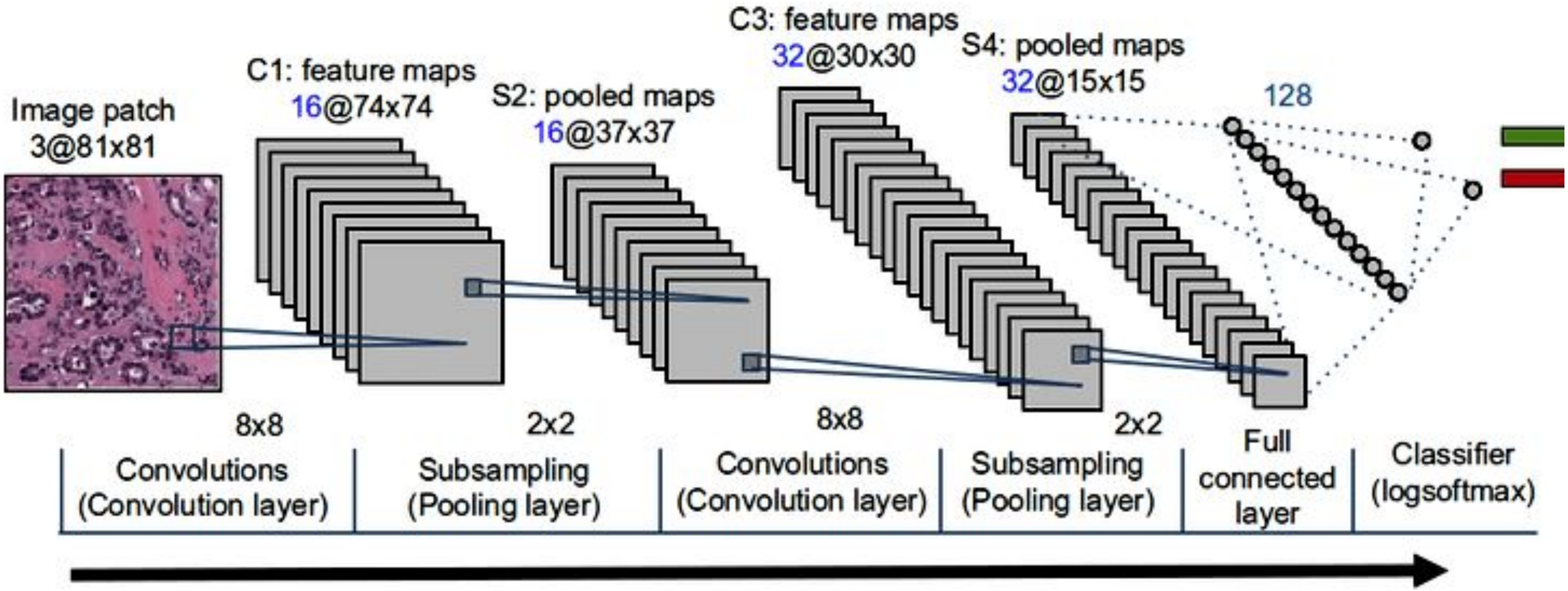
LABORATOIRE
D'INFORMATIQUE
& SYSTÈMES
UMR 7020

- Introduction
  - Description of a convolutional neural network
  - Presentation of the CNN chosen
  - Data sets we used

- Overview of Transfer Learning
  - Fixed Feature Extraction
  - Pre-trained models
  - Fine-tuning
  - Histogram equalization & Data augmentation

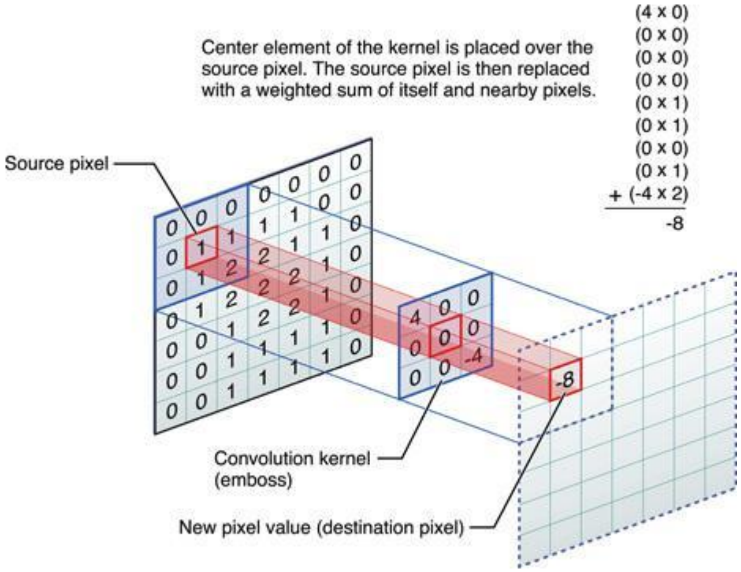
- Final results
- Conclusion

Convolutional Neural Network



3 major parts that compose a CNN

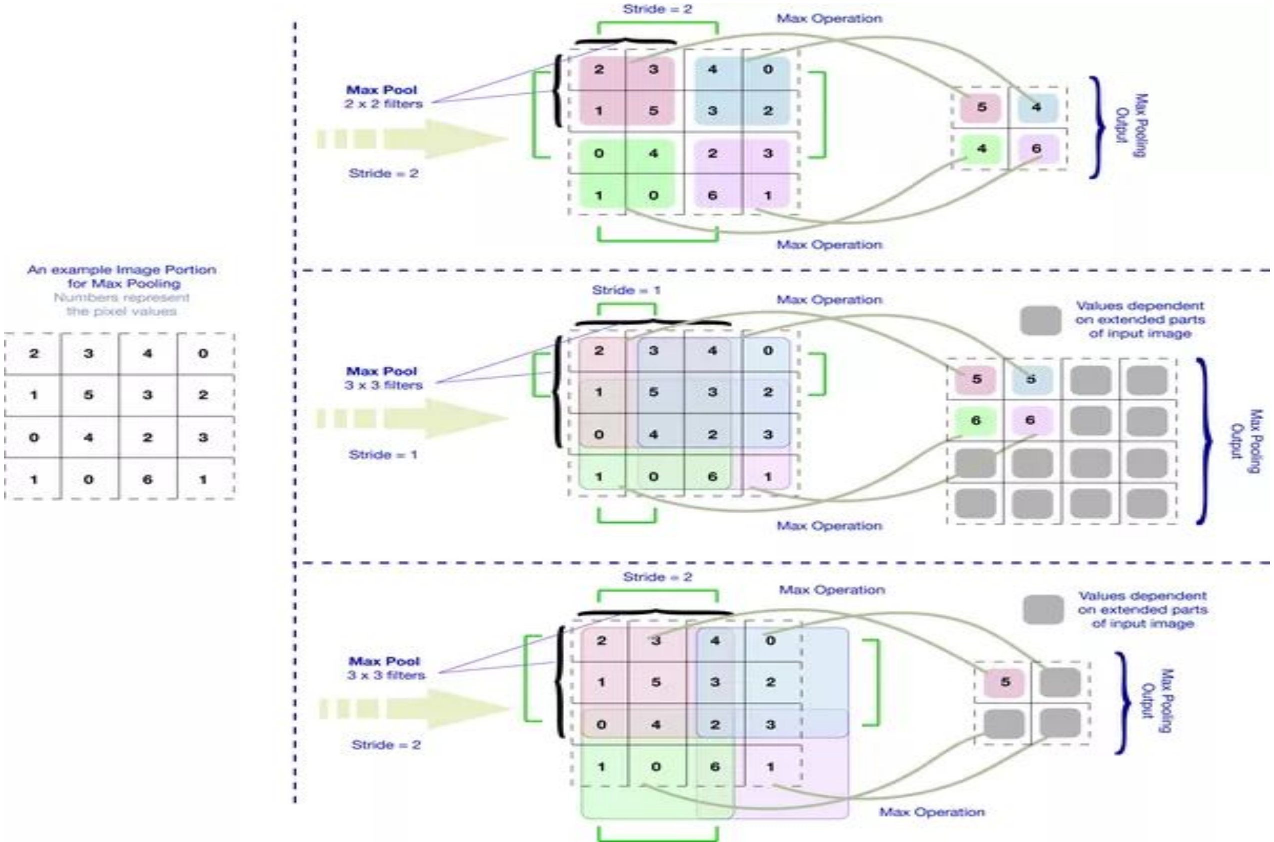
➢ Convolution



Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

3 major parts that compose a CNN

- ➢ Pooling



2	3	4	0
1	5	3	2
0	4	2	3
1	0	6	1

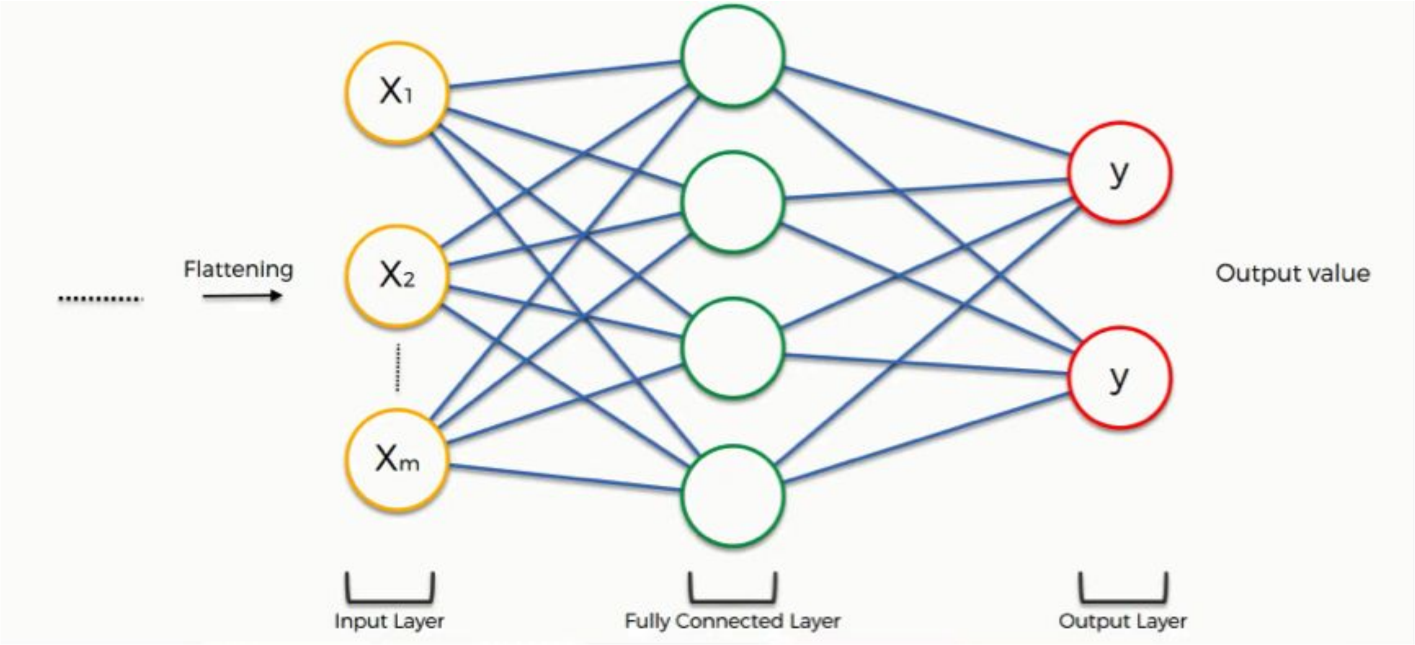
5	4
4	6

5	5		
6	6		

5	

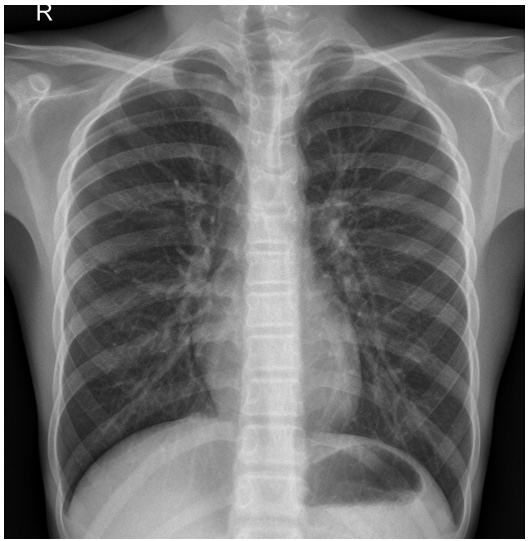
3 major parts that compose a CNN

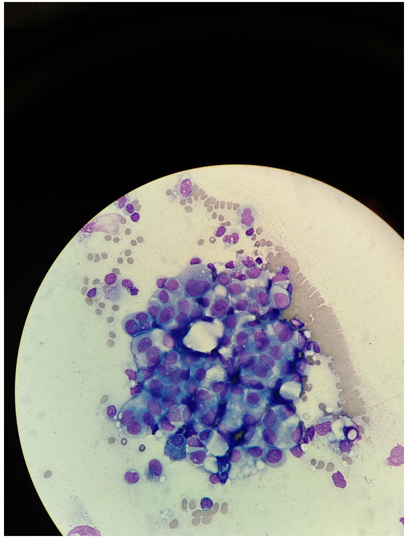
- ➢ Fully connected layer

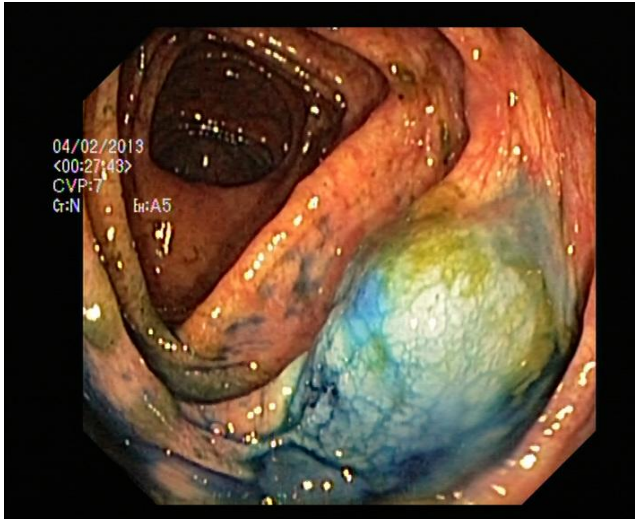


Data sets used

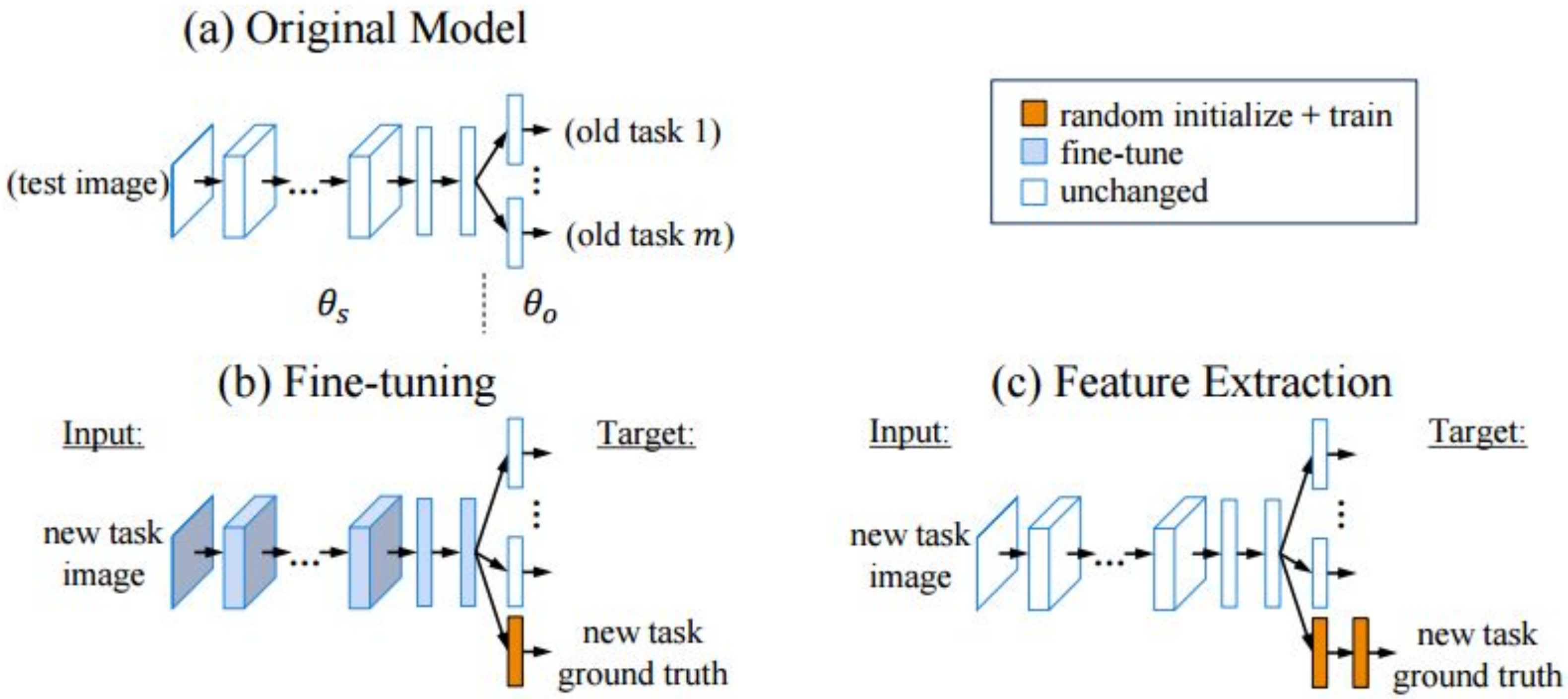
	# of classes	# of images for train, test, val	size & resemblance to imagenet	image size
Chest X-ray (Pneumonia)	2	5221, 624, 16	decent, very far	variable (smallest value is 640)
Cancer cells	2	72, 26, 26	very little, far	4000, 3000
Kvasir v2	8	4800, 1600, 1600	decent, far	720, 576
Mini MIT Etus	3	120, 120, 0	very little, very close	variable (some are close to 128)


R




04/02/2013
<00:27:43>
CVP:7
Gr:N
Eh:A5

Transfert Learning



Fixed Feature Extraction

- CNN codes

- Fully connected layer to fully convolutional layer

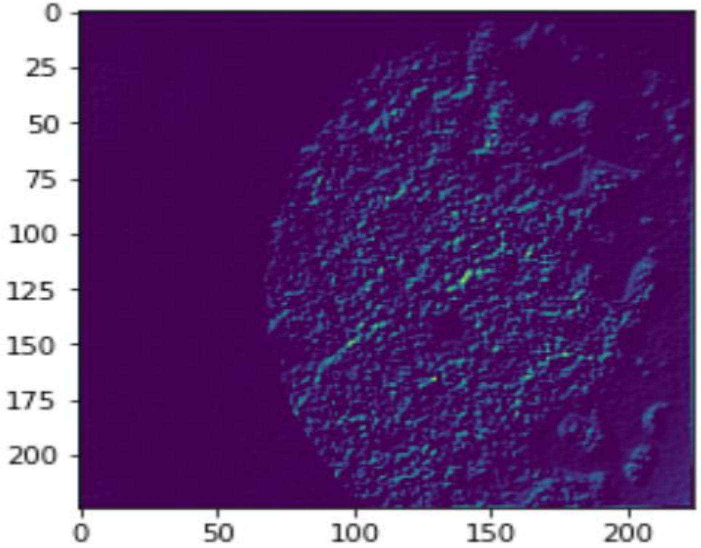
- VGG-19 + SVM

- VGG-19 + VLAD + SVM

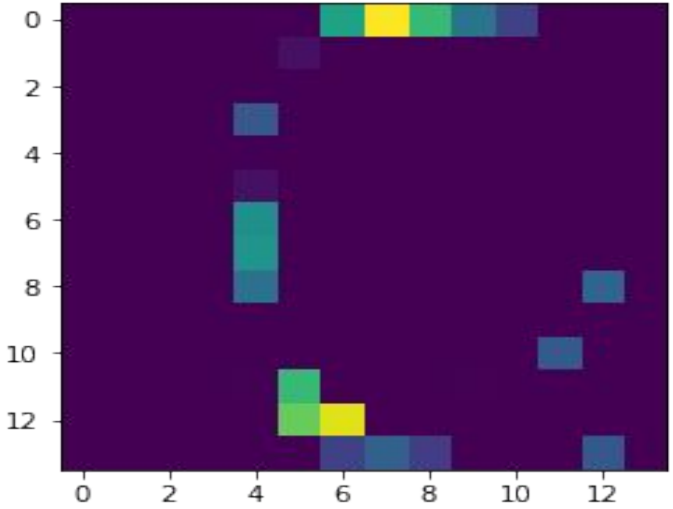
CNN codes

- Assumption : more convolutional layers lead the network to be able to represent more complicated/specified features

block1_conv1



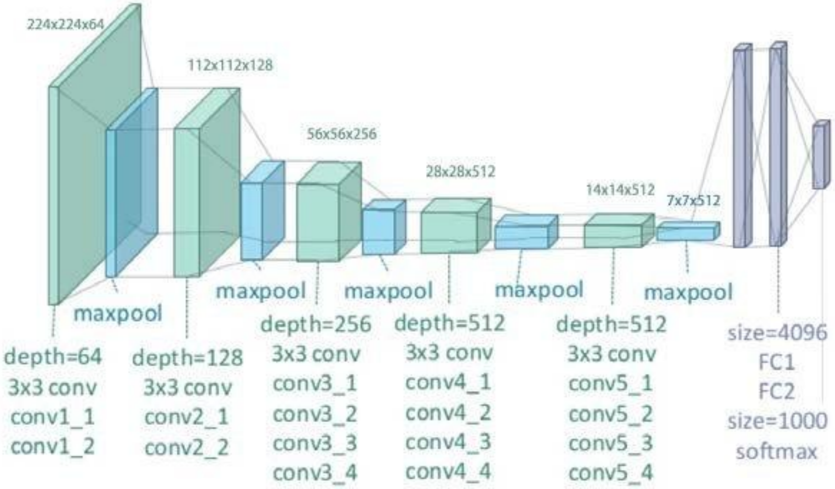
block5_conv1



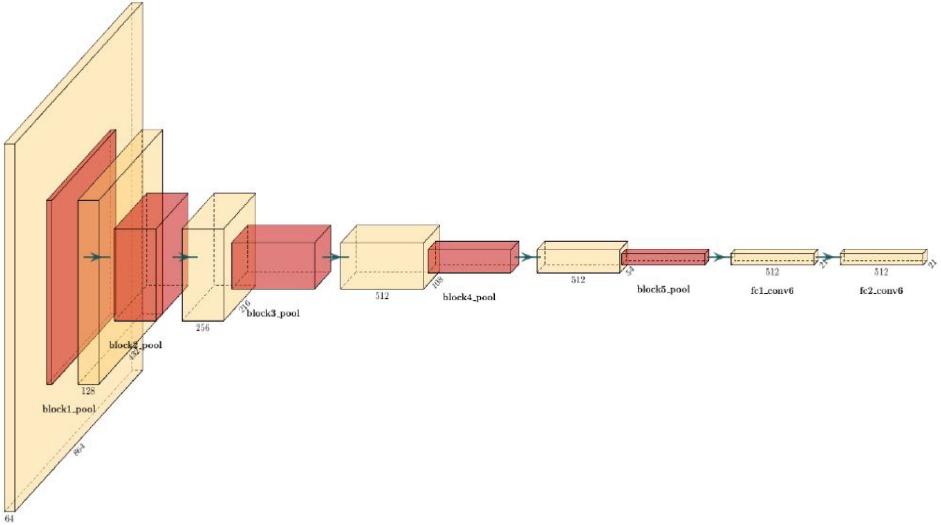
Fully connected layer to fully convolutional layer

- ➢ VGG-19 image size : (224, 224)
- ➢ Wanted image size : (224 + 320n, 224 + 320n), where n = 0, 1, 2

Initial VGG-19



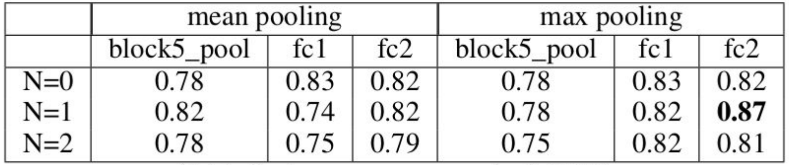
Transformed VGG-19



VGG-19 + SVM

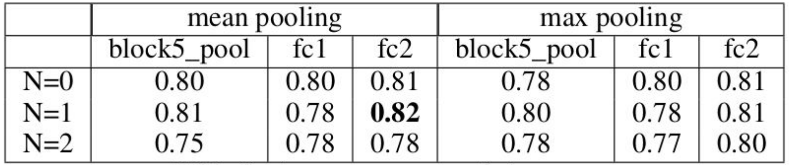
- ➢ Increasing size of input image given to the network matters !

- ➢ n = 1 (224 + 320) seems to be the best parameter nearly always



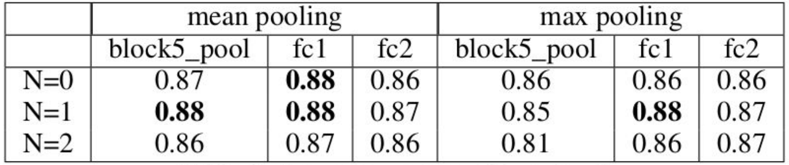
	mean pooling			max pooling		
	block5_pool	fc1	fc2	block5_pool	fc1	fc2
N=0	0.78	0.83	0.82	0.78	0.83	0.82
N=1	0.82	0.74	0.82	0.78	0.82	**0.87**
N=2	0.78	0.75	0.79	0.75	0.82	0.81

Table 1: Accuracy scores of miniMIT



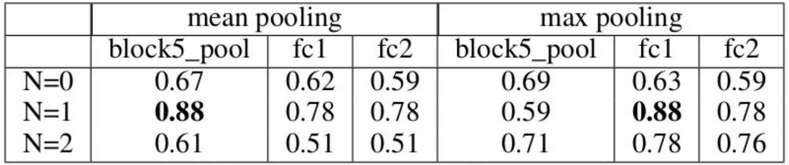
	mean pooling			max pooling		
	block5_pool	fc1	fc2	block5_pool	fc1	fc2
N=0	0.80	0.80	0.81	0.78	0.80	0.81
N=1	0.81	0.78	**0.82**	0.80	0.78	0.81
N=2	0.75	0.78	0.78	0.78	0.77	0.80

Table 2: Accuracy scores of chest_xray



	mean pooling			max pooling		
	block5_pool	fc1	fc2	block5_pool	fc1	fc2
N=0	0.87	**0.88**	0.86	0.86	0.86	0.86
N=1	**0.88**	**0.88**	0.87	0.85	**0.88**	0.87
N=2	0.86	0.87	0.86	0.81	0.86	0.87

Table 3: Accuracy scores of kvasir_v2



	mean pooling			max pooling		
	block5_pool	fc1	fc2	block5_pool	fc1	fc2
N=0	0.67	0.62	0.59	0.69	0.63	0.59
N=1	**0.88**	0.78	0.78	0.59	**0.88**	0.78
N=2	0.61	0.51	0.51	0.71	0.78	0.76

Table 4: Accuracy scores of cancer_cells

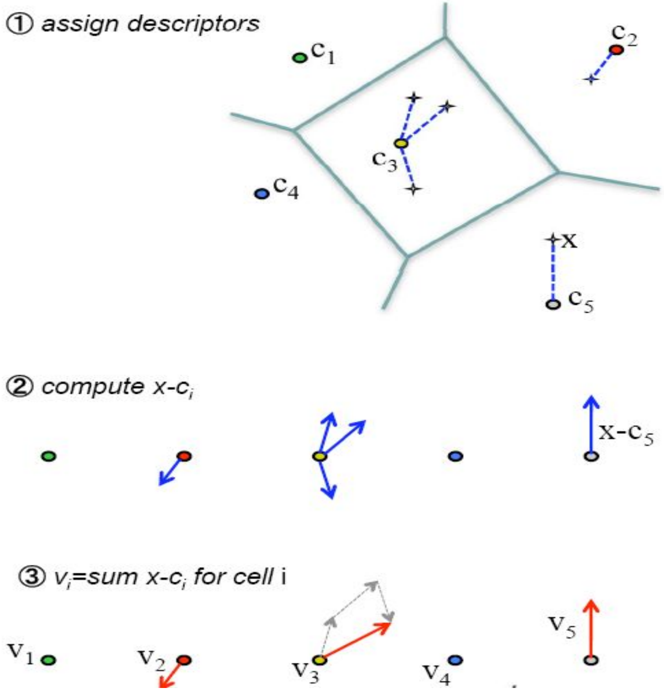
VLAD (Vector of Locally Aggregated Descriptors)

- Learning: *k*-means
  - output: *k* centroids : $c_1,\dots,c_i,\dots c_k$

- VLAD computation:
  - ① $c(x) = \arg\min_{c_i} ||c_i - x||^2$
  - ②③ $v_i = \sum_{x:c(x)=c_i} x - c_i$
  - $v = [v_1, \dots, v_i, \dots, v_k]$, $v_i \in \mathbb{R}^{128}$

  ⇒ dimension D = k *128

- L2-normalized
- Typical parameter: k=64 (D=8192)

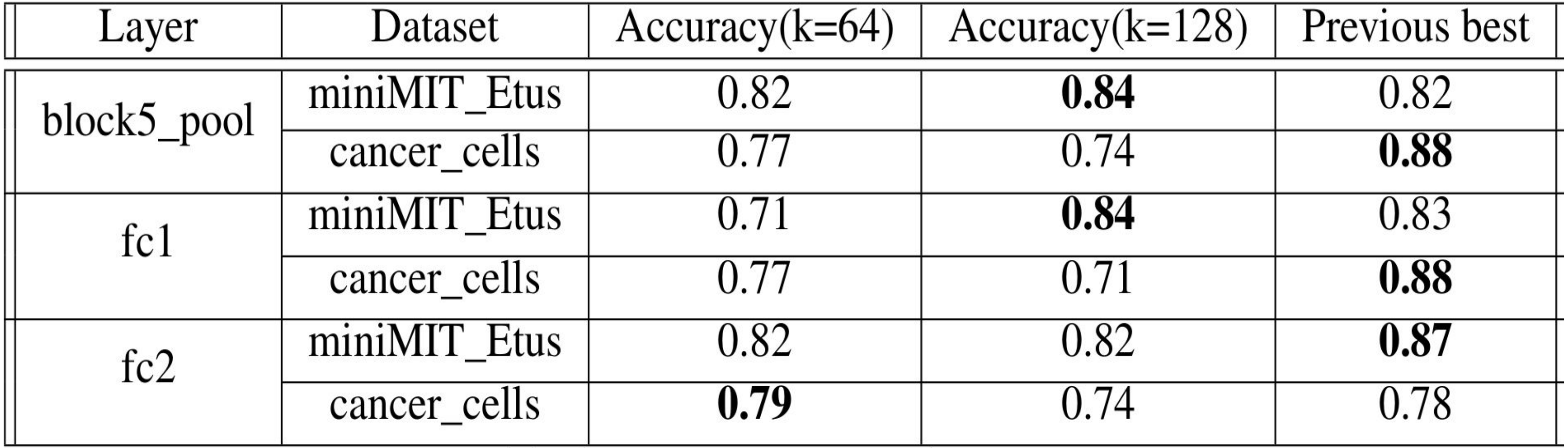


① *assign descriptors*

② *compute x-c_i*

③ *v_i=sum x-c_i for cell* i

VGG19 + VLAD + SVM

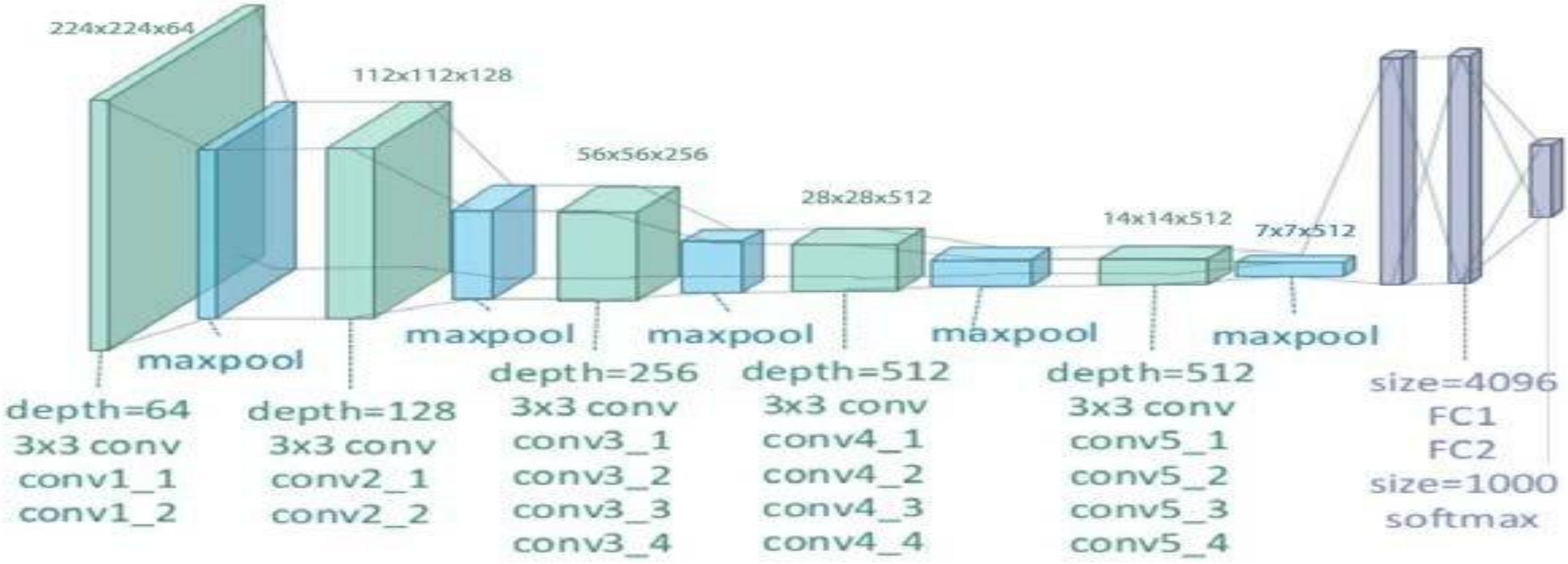


Layer	Dataset	Accuracy(k=64)	Accuracy(k=128)	Previous best
block5_pool	miniMIT_Etus	0.82	**0.84**	0.82
	cancer_cells	0.77	0.74	**0.88**
fc1	miniMIT_Etus	0.71	**0.84**	0.83
	cancer_cells	0.77	0.71	**0.88**
fc2	miniMIT_Etus	0.82	0.82	**0.87**
	cancer_cells	**0.79**	0.74	0.78

- ➢ VLAD doesn’t really increase the performance or slightly

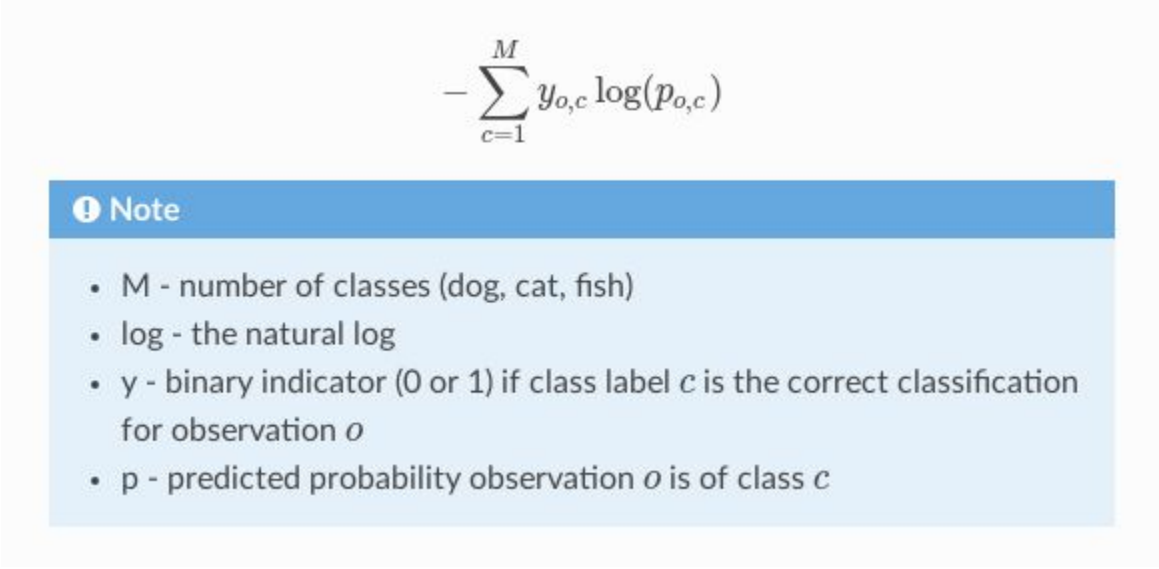
Fine-tuning VGG-19 network

➢ Take a pre-trained model and try to find the best layers to train again



Training parameters

- **300 epochs** with an **early stopping callback** fixed at 20
- Optimize **Adam** with initial learning rate **1e-6**

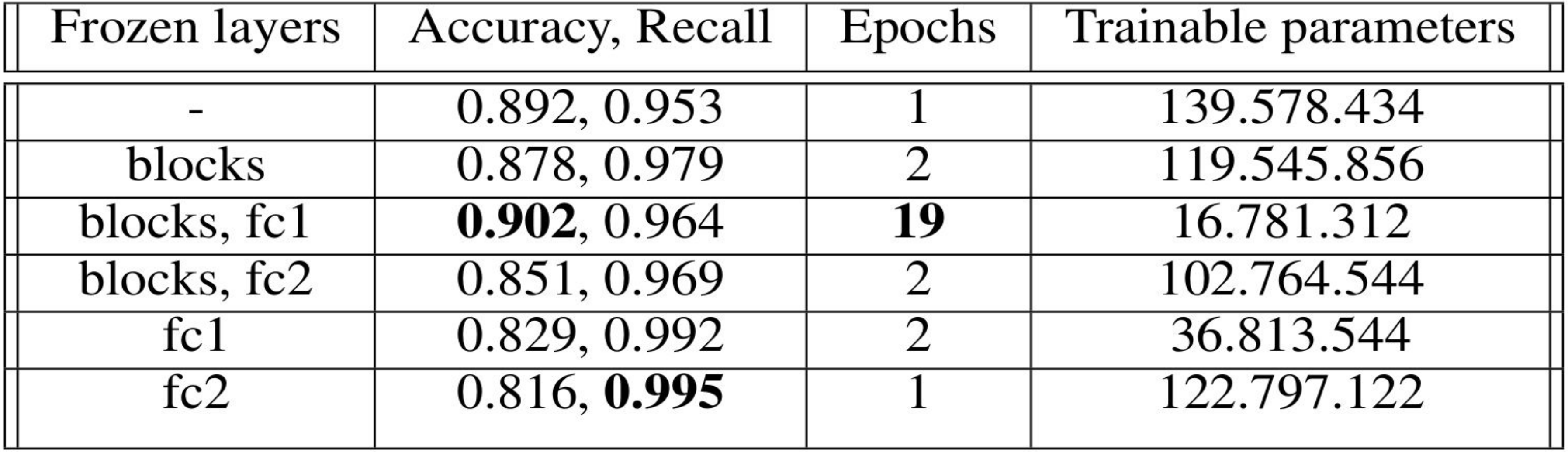


$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Note

- M - number of classes (dog, cat, fish)
- log - the natural log
- y - binary indicator (0 or 1) if class label c is the correct classification for observation o
- p - predicted probability observation o is of class c

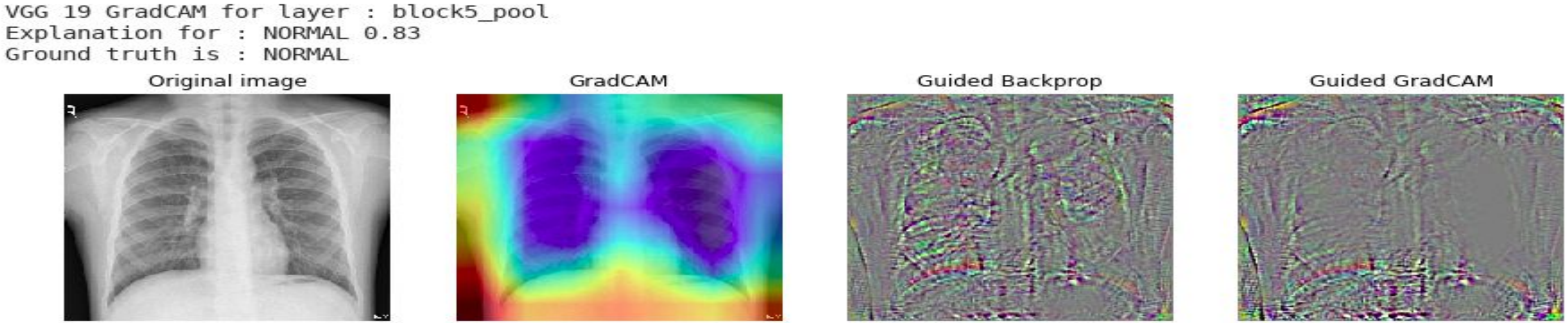
Baseline on Chest X-ray



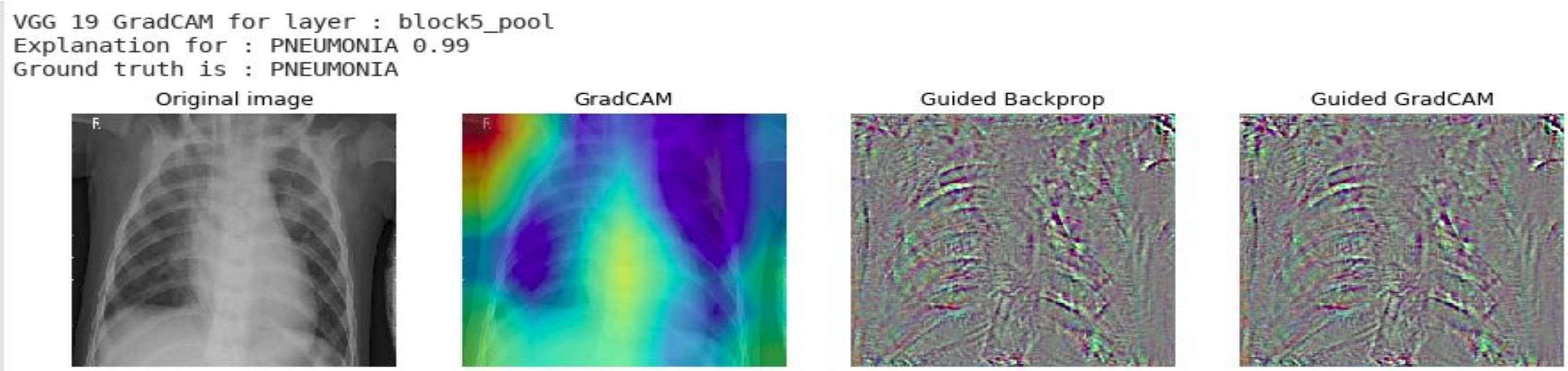
Frozen layers	Accuracy, Recall	Epochs	Trainable parameters
-	0.892, 0.953	1	139.578.434
blocks	0.878, 0.979	2	119.545.856
blocks, fc1	**0.902**, 0.964	**19**	16.781.312
blocks, fc2	0.851, 0.969	2	102.764.544
fc1	0.829, 0.992	2	36.813.544
fc2	0.816, **0.995**	1	122.797.122

- Train the last two layers makes sense !

Let’s try to visualize what we predict

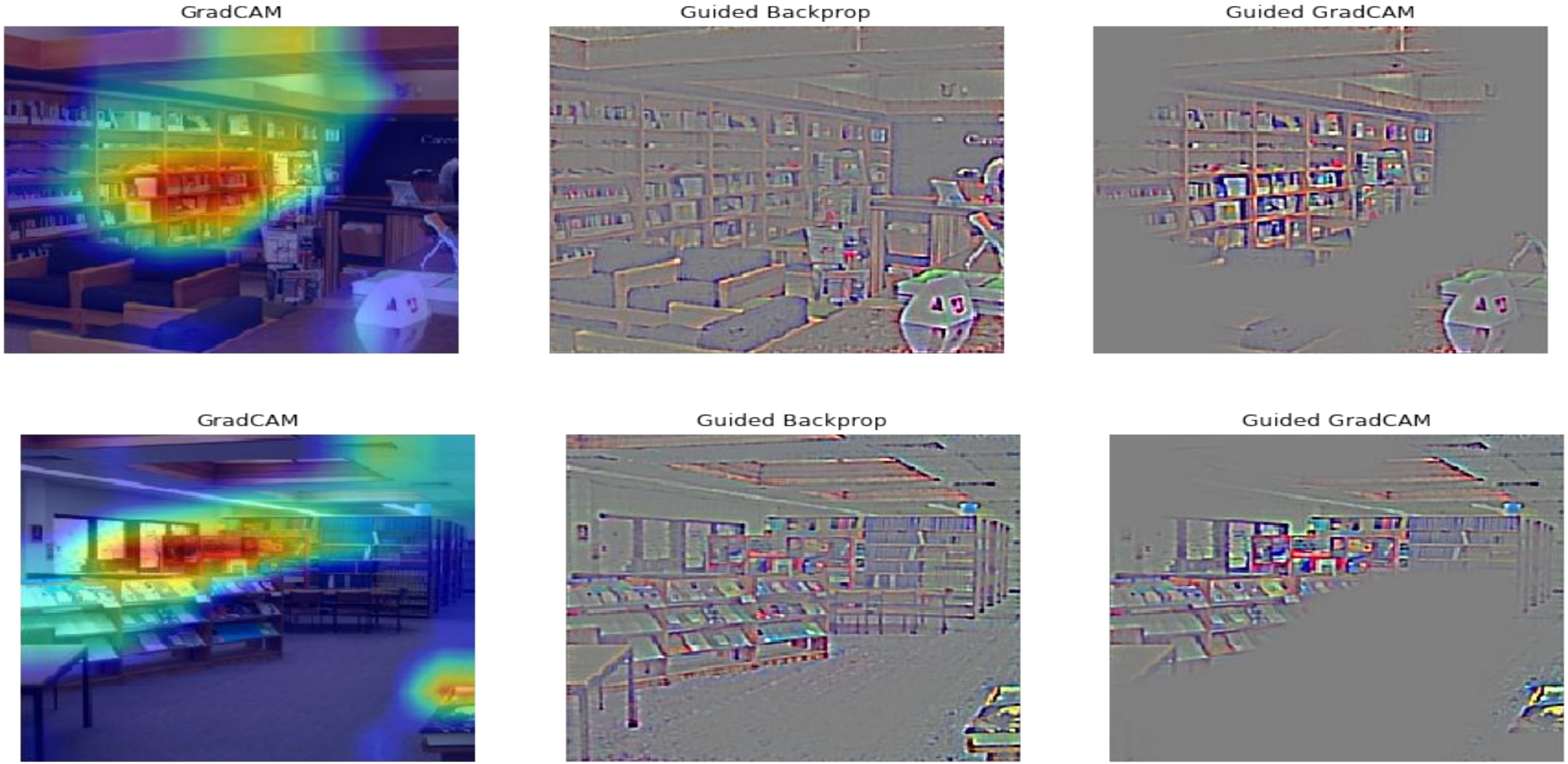


```
VGG 19 GradCAM for layer : block5_pool
Explanation for : NORMAL 0.83
Ground truth is : NORMAL
```



```
VGG 19 GradCAM for layer : block5_pool
Explanation for : PNEUMONIA 0.99
Ground truth is : PNEUMONIA
```

GradCAM comparison : MIT data set (block5_pool)



GradCAM

Guided Backprop

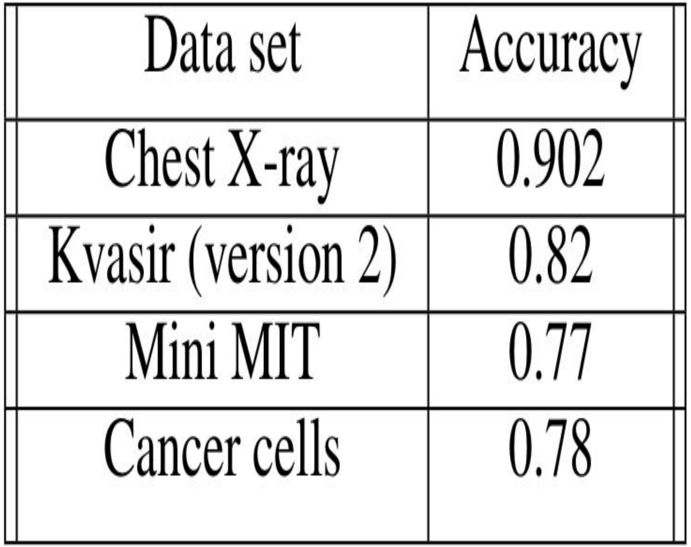
Guided GradCAM

GradCAM

Guided Backprop

Guided GradCAM

What we can conclude at this point

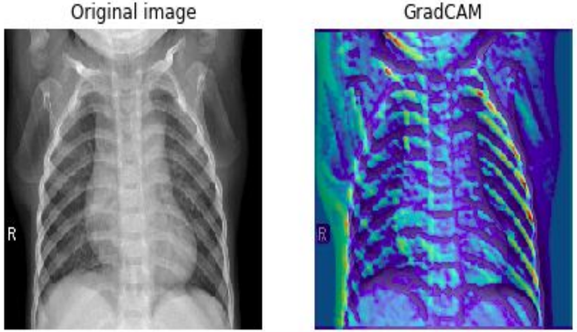


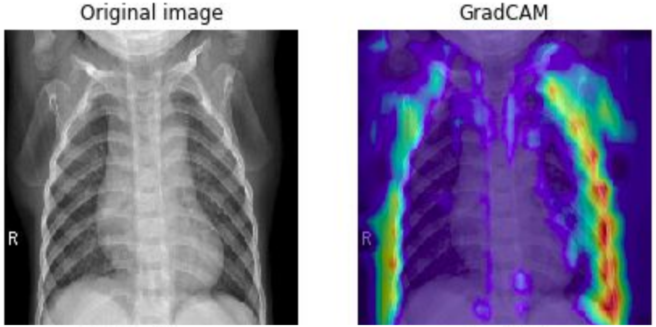
Data set	Accuracy
Chest X-ray	0.902
Kvasir (version 2)	0.82
Mini MIT	0.77
Cancer cells	0.78

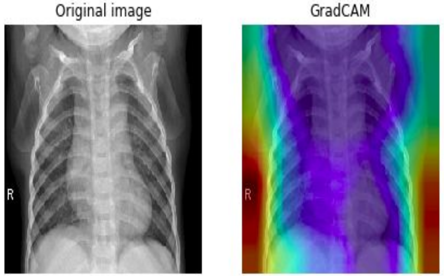
- ➢ Pre-trained models perform better on larger data sets
- ➢ Pre-trained models, even on larger data sets, are overfitted
- ➢ **A network is strong or weak if it can motivate its output**

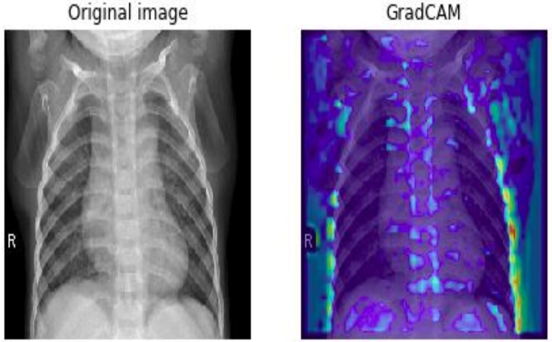
Thus, we decide to explore a way to improve our network in terms of interpretation along to get better metrics results.

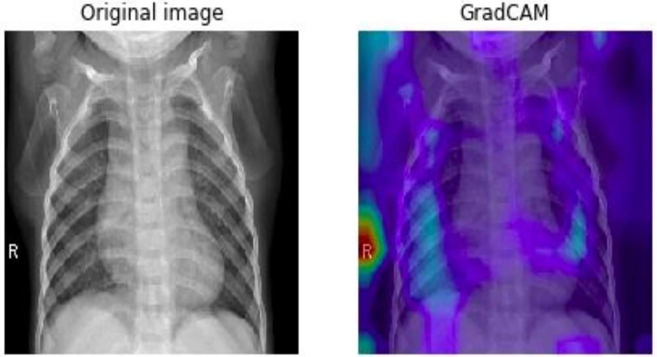
Prediction visualisation for each pool layer







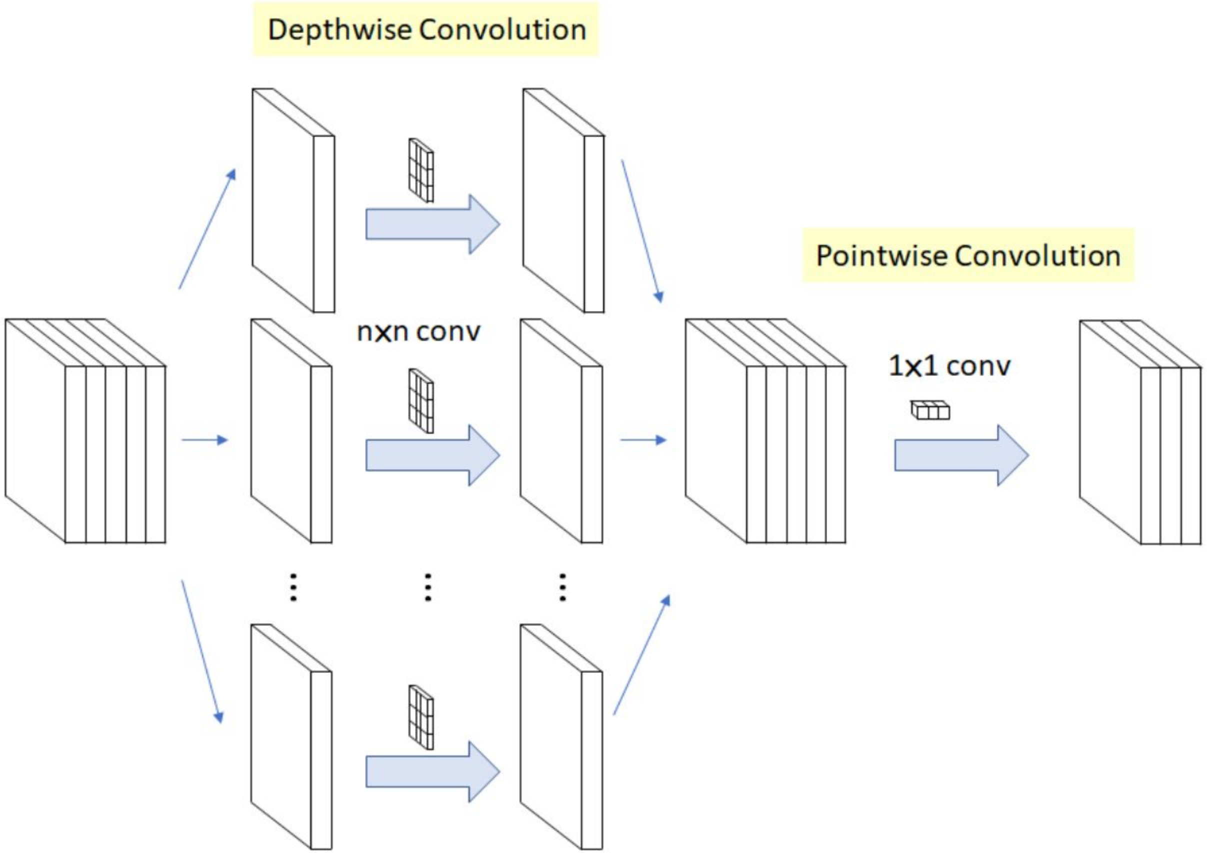




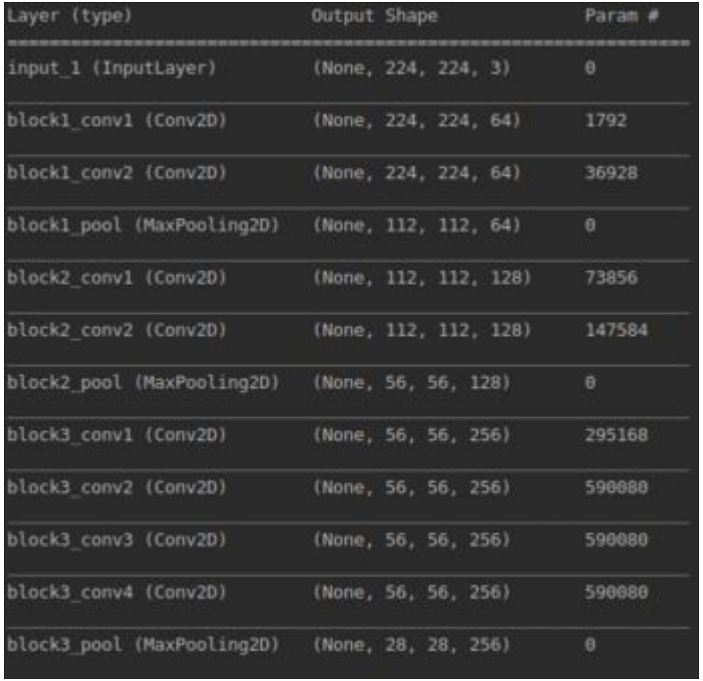
Xception network and depthwise convolutions

- Xception network by François Chollet (creator of Keras)
- Depthwise convolution decrease the computing time with nearly the same result as a normal convolution
  - Main difference is that we transform an image once then elongate it to the number of channels desired
- Based on VGG-19 first 3 blocks, we construct 2 blocks of Depthwise convolution followed by a normalization to prevent overfitting

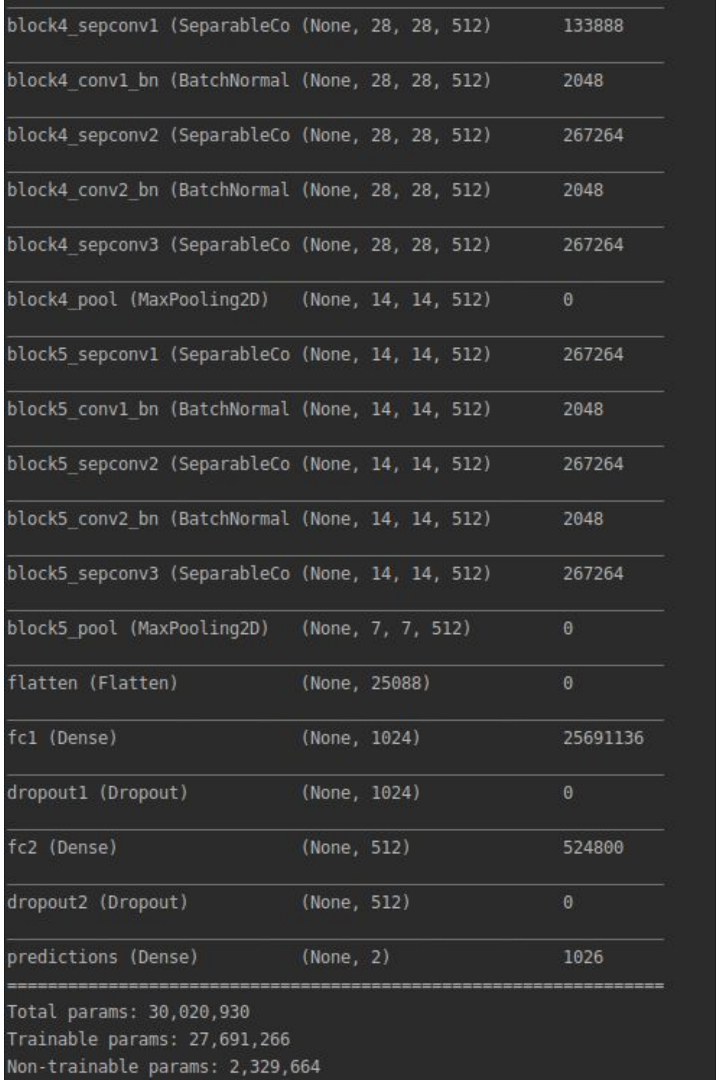
Depthwise convolution



CNN architecture

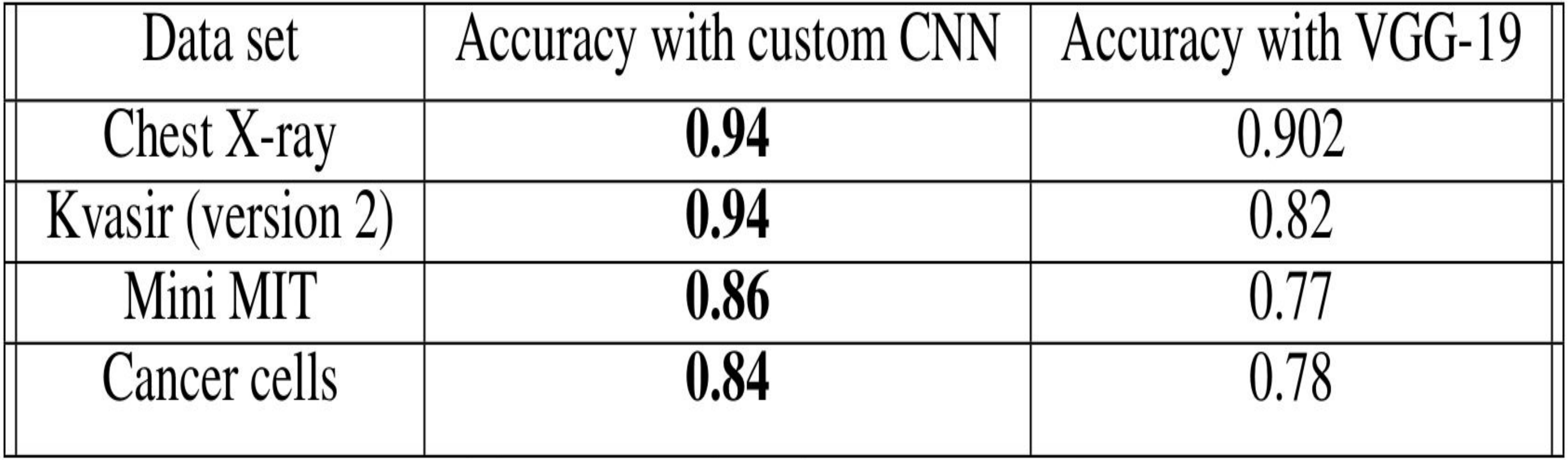


```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
```



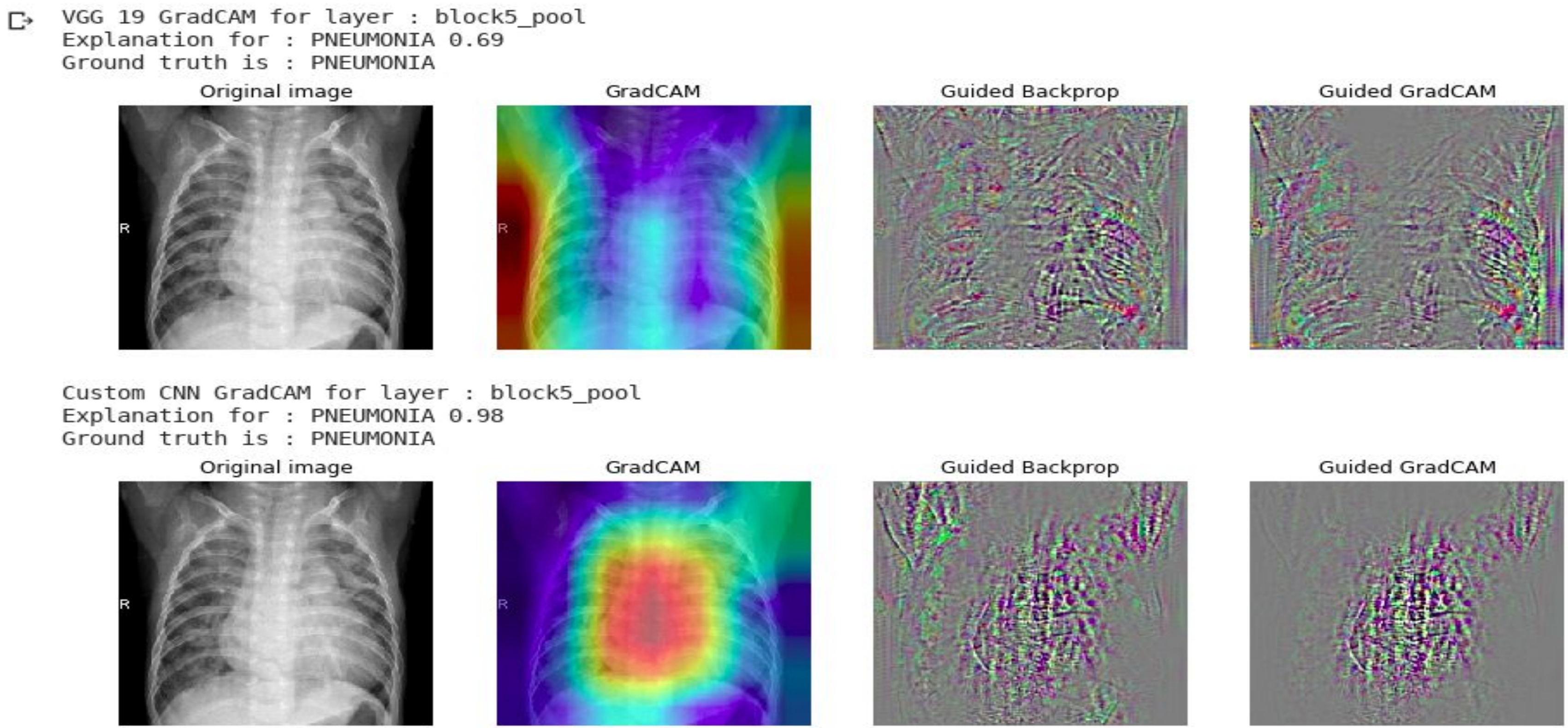
```
_________________________________________________________________
block4_sepconv1 (SeparableCo (None, 28, 28, 512)       133888
_________________________________________________________________
block4_conv1_bn (BatchNormal (None, 28, 28, 512)       2048
_________________________________________________________________
block4_sepconv2 (SeparableCo (None, 28, 28, 512)       267264
_________________________________________________________________
block4_conv2_bn (BatchNormal (None, 28, 28, 512)       2048
_________________________________________________________________
block4_sepconv3 (SeparableCo (None, 28, 28, 512)       267264
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_sepconv1 (SeparableCo (None, 14, 14, 512)       267264
_________________________________________________________________
block5_conv1_bn (BatchNormal (None, 14, 14, 512)       2048
_________________________________________________________________
block5_sepconv2 (SeparableCo (None, 14, 14, 512)       267264
_________________________________________________________________
block5_conv2_bn (BatchNormal (None, 14, 14, 512)       2048
_________________________________________________________________
block5_sepconv3 (SeparableCo (None, 14, 14, 512)       267264
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 1024)              25691136
_________________________________________________________________
dropout1 (Dropout)           (None, 1024)              0
_________________________________________________________________
fc2 (Dense)                  (None, 512)               524800
_________________________________________________________________
dropout2 (Dropout)           (None, 512)               0
_________________________________________________________________
predictions (Dense)          (None, 2)                 1026
=================================================================
Total params: 30,020,930
Trainable params: 27,691,266
Non-trainable params: 2,329,664
```

Results obtained with that CNN



Data set	Accuracy with custom CNN	Accuracy with VGG-19
Chest X-ray	**0.94**	0.902
Kvasir (version 2)	**0.94**	0.82
Mini MIT	**0.86**	0.77
Cancer cells	**0.84**	0.78

Prediction visualisation for last conv layer



```
VGG 19 GradCAM for layer : block5_pool
Explanation for : PNEUMONIA 0.69
Ground truth is : PNEUMONIA
```

```
Custom CNN GradCAM for layer : block5_pool
Explanation for : PNEUMONIA 0.98
Ground truth is : PNEUMONIA
```

Other tricks we used

- ➢ Pixel regularization
  - ○ no improvement
  - ○ may be combined with raw image

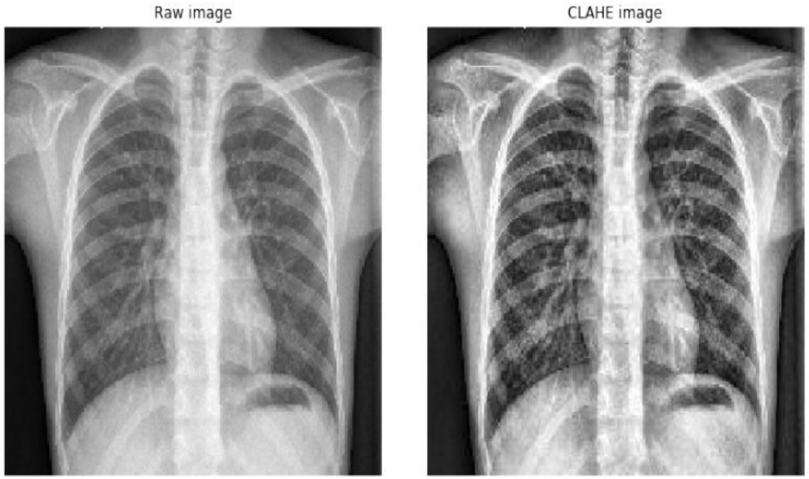
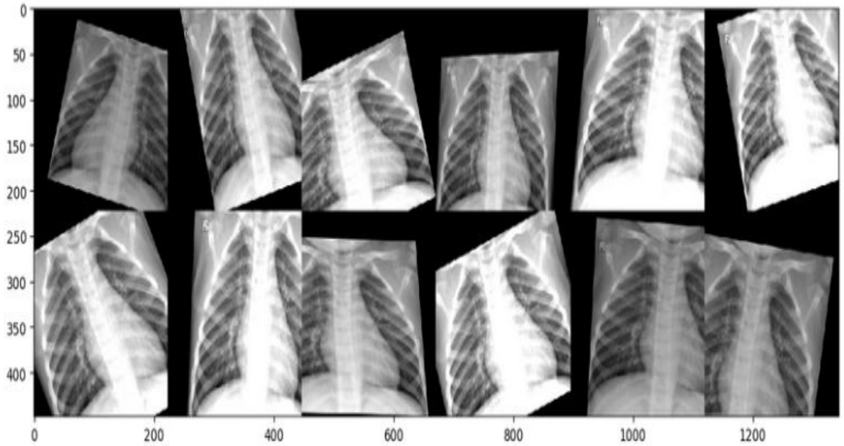
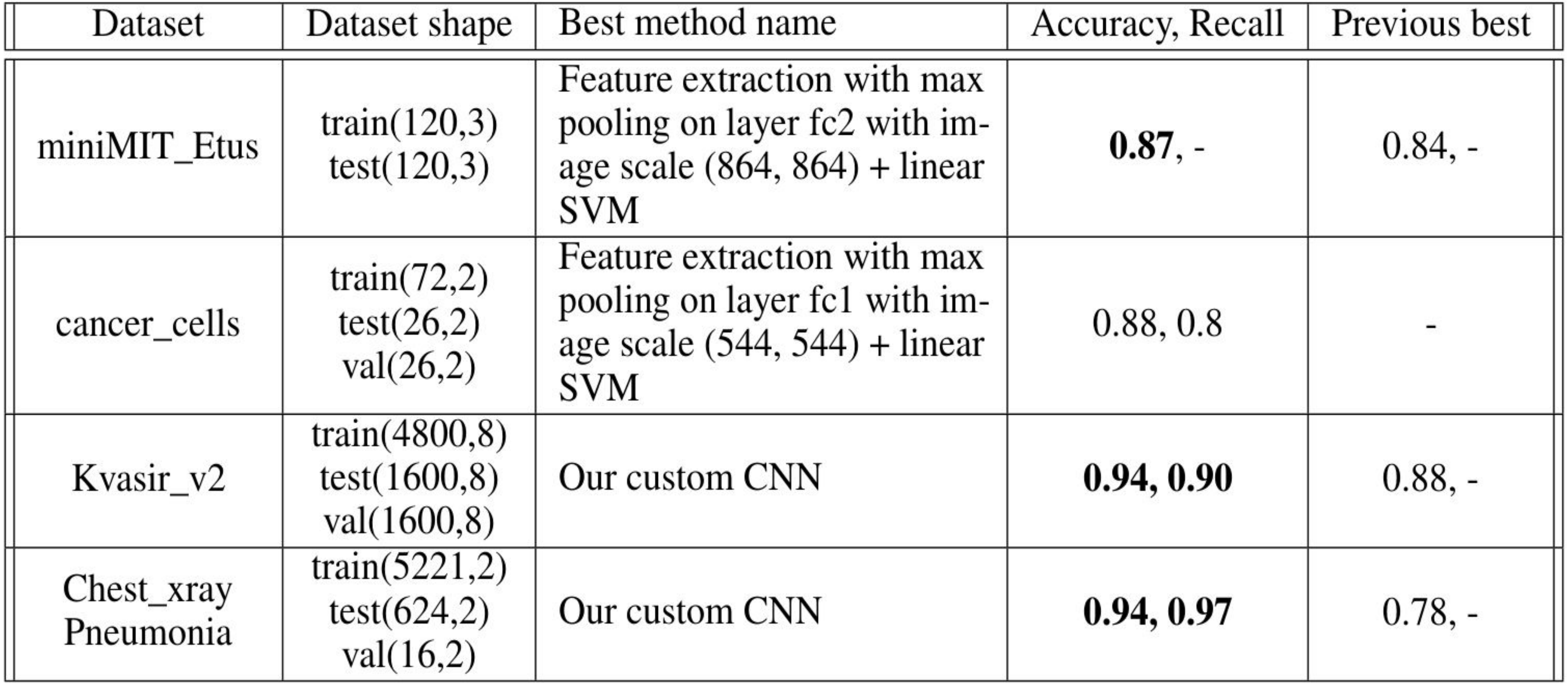


Figure 33: Difference between a raw and normalized image from Chest X-ray data set

- ➢ Data augmentation
  - ○ good improvement for smaller datasets (5~10%)



Final results



Dataset	Dataset shape	Best method name	Accuracy, Recall	Previous best

miniMIT_Etus	train(120,3)
	test(120,3)
		Feature extraction with max pooling on layer fc2 with image scale (864, 864) + linear SVM	**0.87**, -	0.84, -

cancer_cells	train(72,2)
	test(26,2)
	val(26,2)
		Feature extraction with max pooling on layer fc1 with image scale (544, 544) + linear SVM	0.88, 0.8	-

Kvasir_v2	train(4800,8)
	test(1600,8)
	val(1600,8)
		Our custom CNN	**0.94, 0.90**	0.88, -

Chest_xray Pneumonia	train(5221,2)
	test(624,2)
	val(16,2)
		Our custom CNN	**0.94, 0.97**	0.78, -

Conclusion

- 4 data sets of various properties
  - Transfer Learning scenarios to use for a image classification project
- **Size of the data set and its similarity to the original data set matters**
  - **Large** ⇒ **fine-tuning** of a pre-trained network
  - **Small** ⇒ **linear classifier** on fully connected layers from a pre-trained network.
  - **VLAD** ⇒ **not so relevant** but could improve the accuracy from the last convolutional layer
- In case the data set you use is very similar to the ImageNet data set, you should just use the best pre-trained network available nowadays
- Excellent neural network in regards of metrics NOT always useful
  - Output should be relevant for humans.