DEEP LEARNING APPLIED TO MEDICAL IMAGES RECOGNITION

Chen Dang*
Department of Computer Science
Aix-Marseille University
163 Avenue de Luminy, 13009 Marseille
chen.dang@etu.univ-amu.fr

Hippolyte L. Debernardi*
Department of Computer Science
Aix-Marseille University
163 Avenue de Luminy, 13009 Marseille
contact@hippolyte-debernardi.com

Ronan Sicre
Department of Computer Science
Centrale Marseille
38 Rue Frédéric Joliot Curie, 13013 Marseille
ronan.sicre@lis-lab.com

July 11, 2019

ABSTRACT

Since Alexey Ivakhnenko and Lapa introduced the multilayer perceptrons in 1965, methods based on deep learning has been significantly improved, and especially within the field we are interested in, image classification problem. This report is about the medical images classification problem. Image classification consists of knowing the category of an input image, based on the output of a system trained in that purpose, thanks to an annotated by human image set. That process can be applied to many fields, such as the very famous ImageNet designed to recognize an input image between a tremendous 21.841 categories set based on the learning of millions of images.

We will focus on the use of Convolutional Neural Networks to diagnosis medical images. Our experiments were conducted on the field of Transfer Learning.

Firstly, our work covers the extraction of the features produced by the processing of an image inside a pre-trained CNN to feed a Support Vector Machine, use of Principal Component Analysis and Vector of Locally Aggregated Descriptors to improve performance, which is called fixed features extractor.

In a second time, we will explore pre-trained models and fine-tuning two different networks, VGG-19 and Xception. During that process, we will put a special attention to visualize what our network predict.

We finally use some image processing methods such as pixel regularization and data augmentation in order to see the evolution of previously listed methods with increased amount of data.

Keywords Deep Learning · Medical images recognition · Transfer Learning

Contents

1 Deep learning and machine learning methods

Deep learning is one of the machine learning technique that learns features directly from data. When the amount of data is increased, deep learning usually gives better performance. It's usually hard to answer on the 'What is amount of big' question, and we'll see later on various data set sizes. Today, deep learning is mainly used in speech recognition, image classification, natural language processing or recommendation systems.

The main differences of deep learning from machine learning are :

- machine learning covers deep learning.
- features are given to machine learning manually.
- deep learning learns features directly from data.

Due to the inherence of deep learning, it's sometimes hard to get a feedback on what your network learns. We will try to verify what we predict as our experiments continue.

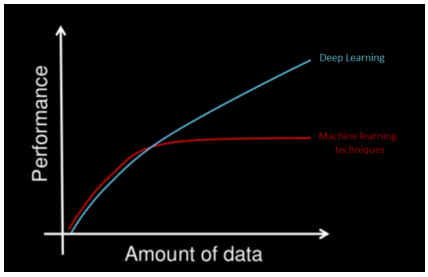

Figure 1: Performance evolution over time

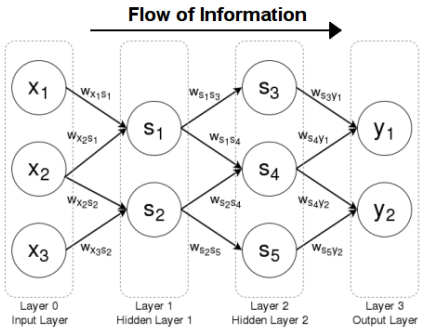

Figure 2: A Neural Network overview

1.1 Convolutional Neural Networks

A Convolutional Neural Network - CNN - is an artificial neural network with multiple layers of so-called convolutions. To have a global view on how it works, we can see it as a big machine that takes an image as input, which is then processed by the different layers of the network, and finally outputs probabilities. These probabilities could be anything, so you can predict various things, from facial to cancer recognition.

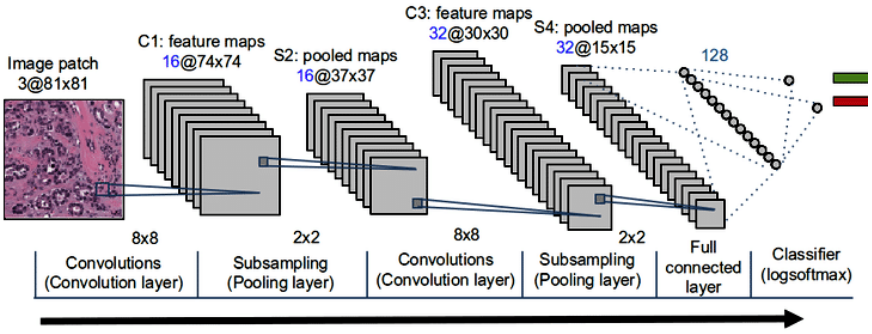

Figure 3: A convolutional Neural Network overview

Such a network contains three major parts : convolutions, pooling and fully connected layers.

1.1.1 Convolution

A convolution consists as a local analyze : it does a convolution product between the input image (h x w x d) and a filter (fh x fw x d) to return a map of size ((h - fh +1) x (w - fw +1) x 1) if there is no padding.

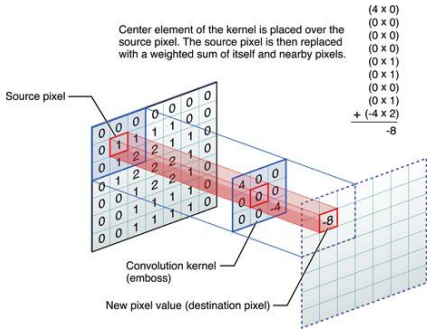

Figure 4: Performance evolution over time

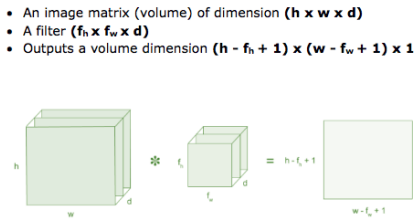

Figure 5: Applying a convolution to a volume

1.1.2 Pooling

A pooling take a convolution map (h x w) and apply it a mathematical function (min, max, average) to get a new map.

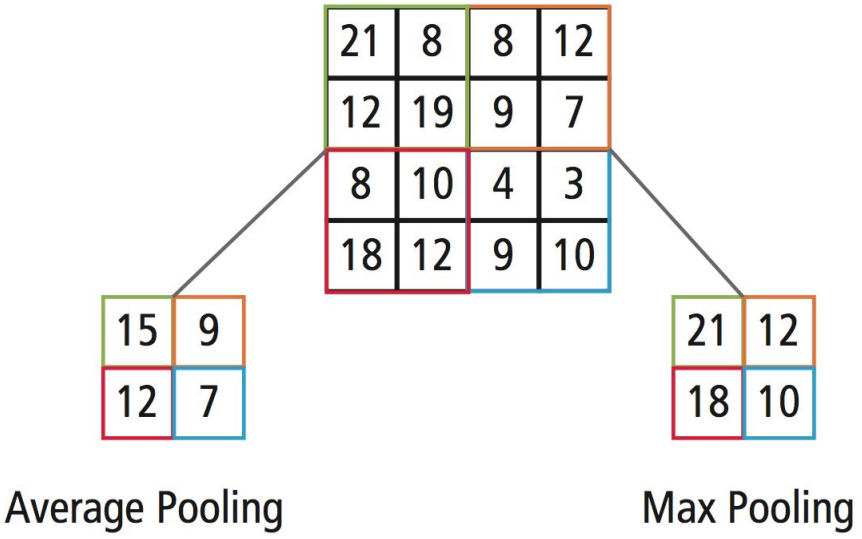

Figure 6: Average and Max Pooling

1.1.3 Fully Connected Layer

A fully connected layer take the output array of a previous layer and perform a pooling on the different elements by performing operations with all of them.

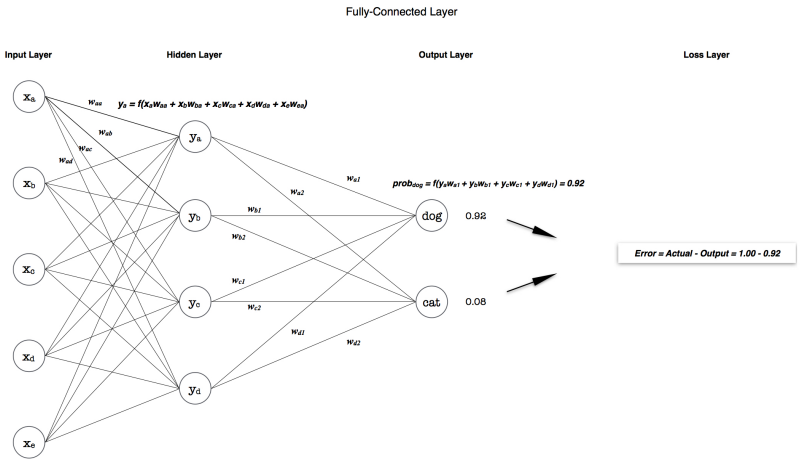

Figure 7: Fully Connected Layer overview

2 Previous work done

This work is a follow-up to the works of two other students : Hichma KARI and Houssem FARHAT.

2.1 H. Kari

The work of H. KARI shows that the features extracted from convolutional neural network can be used to classify medical images with a linear SVM. A VGG-19 convolutional neural network pre-trained on ImageNet was used. Features are generated from three last layers of network: block5, fc1 and fc2. The impact of different scales of images and different pooling methods are also studied. The result showed a good prediction on the datasets. His work also mentioned the possibility of using aggregation to extract image features and other networks, which may improve the performance of the classification.

2.2 H. Farhat

H. FARHAT's work studied the combination of CNN and Vector of locally aggregated descriptors (VLAD) aggregation used for medical image classification. A VGG-19 CNN pre-trained on ImageNet is also used to extract features from images and the output layer is block5_pool. The features generated from the same image of three different scales are combined and then reduced to a dimension of 128 with principal component analysis(PCA). Afterwards, VLAD is applied and a linear SVM using the result produced is employed to make the classification. This method didn't have a remarkable improvement for all the datasets. This work also proposed to use different layer of VGG-19 to extract features or different CNN such as Resnet.

3 Data sets used

3.1 Chest X-ray

This data set present chest x-ray images selected from pediatric patients of one to five years old from Guangzhou Women and Children Medical Center, at Guangzhou. The data set contains 5,863 chest x-ray images which can be classify into two categories : those of pneumonia patients and those of healthy patients. The data is split into three folders, a train set, a test set and a validation set, and each of them contains two folders for the two categories. However, we didn't use the validation set for this classification since it only contains 16 images.

The data set can be download there :

`https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia`

You can find the state-of-art for this data set there :

`https://arxiv.org/abs/1711.05225`

`https://www.cell.com/cell/fhttps://www.cell.com/cell/fulltext/S0092-8674(18)30154-5ulltext/S0092-8674(18)30154-5#fig6`

3.2 Kvasir (version 2)

The data is collected using endoscopic equipment at 4 hospitals in Norway and are carefully annotated by one or more medical experts. The kvasir-dataset-v2 archive contains 8,000 images, 8 classes, 1,000 images for each class. The classes are Z-line, pylorus, cecum, esophagitis, polyps, ulcerative colitis, dyed and lifted polyps and dyed resection margins. We split the dataset into three folders: a train set, a test set and a validation set, containing 4800, 1600 and 1600 images respectively.

The detailed description of the dataset and the download link is here:

`https://datasets.simula.no/kvasir/`

3.3 Mini MIT

The Mini MIT data set is a reduced data set from the MIT 67 scenes data set. It contains two sets: a train set and a test set, having 40 images in each category of each set. The categories are bookstore, library and inside_bus.

The dataset is downloaded here:

`https://drive.google.com/drive/folders/1VvTmisB4caF7Ux0rcJo2ippllagKk20K`

3.4 Cancer cells

Cancer_cells is a data set containing 123 microscope photos of cancer cells and 121 microscope photos of non-cancer cell,collecting from a partner hospital of LIS laboratory. Each image has 4000 * 3000 pixels and annotated by medical experts. We split the data set into three parts: a train set, a test set and a validation set, containing 145, 51 and 48 images respectively.

4 Our experiences and works

In practice, it's rare to train an entire ConvNet with random initialization because it requires a very large data set (probably millions of training images for a network as deep as VGG-19). Instead, it's more common to use a pre-trained network either as an initialization to **fine-tune** another model or as a **fixed feature extractor** for the task of interest. Such an approach is called **Transfer Learning**. We use three of transfer learning scenarios presented on figure 9 based mainly on the VGG-19 network which detailed structure is shown of the figure 8.

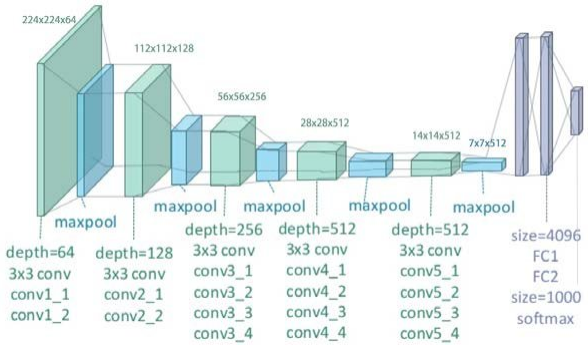

Figure 8: VGG-19 overview

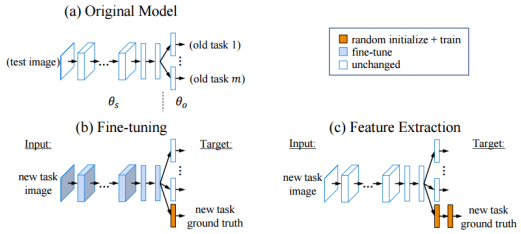

Figure 9: Fine-tuning

4.1 Fixed Feature Extraction

In this step we use the pre-trained on ImageNet dataset neural network called VGG-19 as a feature extractor. This network is composed with five blocks of convolutional layers and three fully connected layers.

Each layer of the convolutional layers represent a certain level of graphic feature. In general, the more convolutional layers we have, the more complicated features can be represented. For example, the first convolutional layer may detect borders of different directions, and the next layer may detect all the combinations of borders. In this way, the last convolutional layer may be able to represent very complicated image features. These features are send to several fully connected layers to produce more complicated features and finally make the prediction.

For example, the image of a cancer cell and the result correspondent of one of the filter in the first layer block1_conv1 of VGG-19 are shown as figures below:

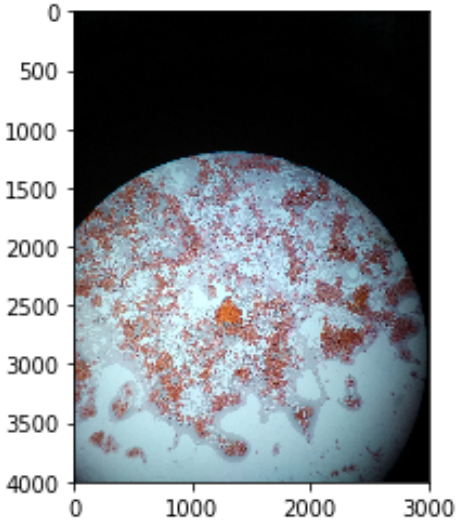

Figure 10: cancer cell microscope photo

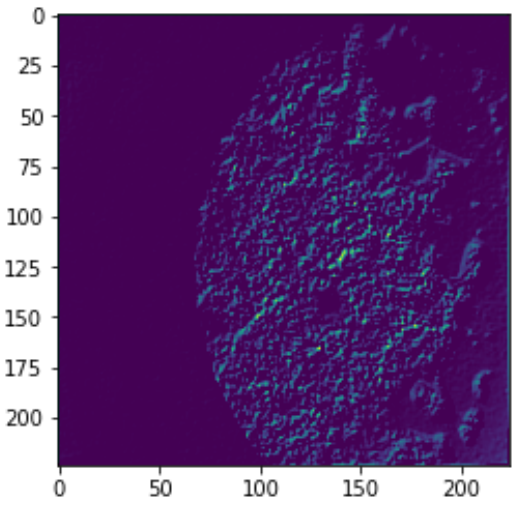

Figure 11: feature extracted

The features of different layers extracted from this image are shown as figures follow:

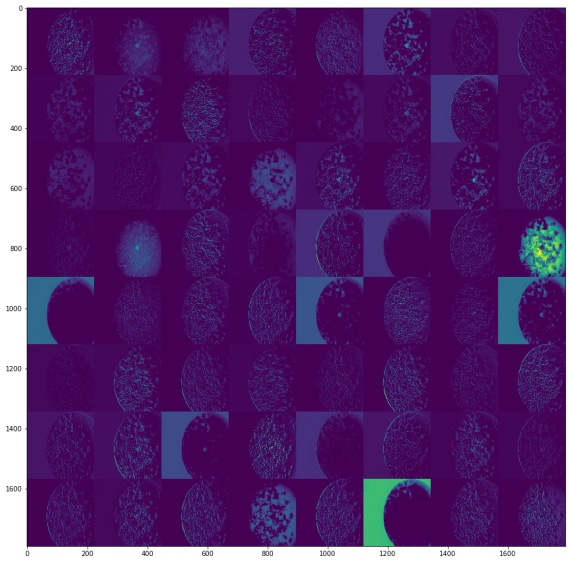

Figure 12: features from block1_conv1

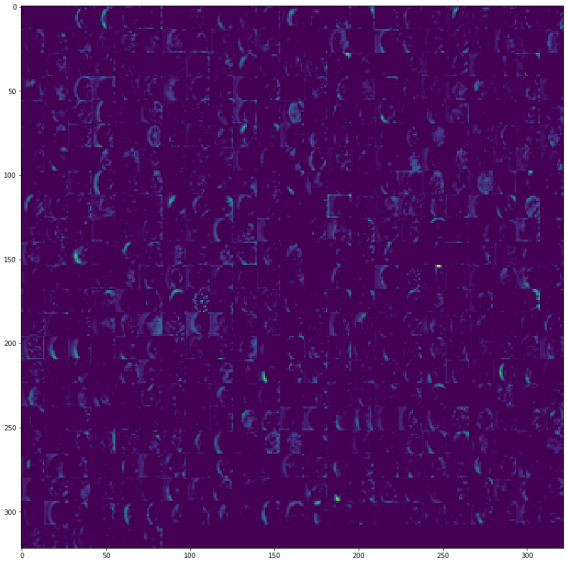

Figure 13: features from block5_conv1

Previous work has shown that features of different scales images extracted from block5_pool layer of VGG-19 can have good result with linear kernel SVM. The sizes of images are $(224 + 320n, 224 + 320n, 3)$, where n is the scale, which is among 0, 1 and 2. An average pooling and the L2 normalization are applied to the features. An SVM are then used for the classification. In our work, we want to know the impact of changing layer of feature extraction and different pooling methods. We tried max pooling and average pooling on the features extracted from last three layers of VGG-19: block5_pool, fc1 and fc2.

VGG-19 is designed for the images of size $(224, 224, 3)$. To get features correctly of different image scales, the last two layers of fully connected must be changed to convolutional layers. The weights of fc1 and fc2 trained from ImageNet are then used as a filter to be applied to the features. This is therefore a fully convolutional neural network. For example, the neural network structure when image scale is 2 is shown as figure follow:

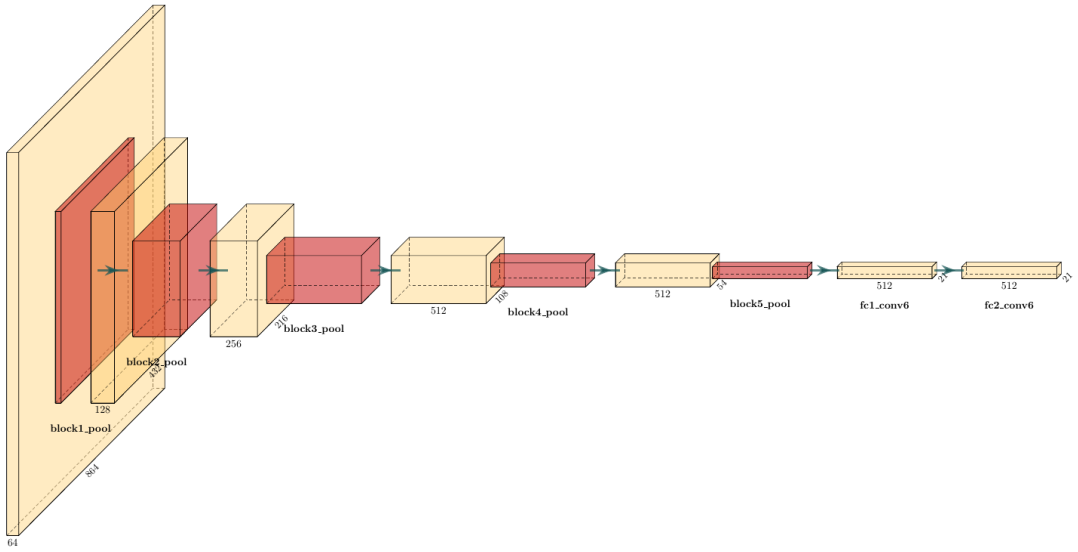

Figure 14: Fully convolutional CNN structure

The data sets we used in this step is as before: miniMIT, chest_xray, kvasir_v2 as well as cancer_cells. The results are in the tables below:

	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

The result shows that when we extract features from fc1 layer or fc2 layer, max pooling almost always gives higher accuracy score than mean pooling. While extracting features from block5_pool, mean pooling is a better choice.

We have also notice that in some small data sets, such as miniMIT_Etus and cancer_cells, extracting features from two last layers can have significant improvement to the accuracy score. However when the size of data set is relatively large, the influence of extracting features from different layers become not so obvious.

To give an intuitive representation of the prediction, we used GradCAM to visualize what we have extracted from the layer block5_pool. GradCAM, which means Gradient-weighted Class Activation Mapping, can produce a coarse localization map highlighting the important regions in the image for predicting. It uses the gradients of a image, flowing into certain convolutional layer. The advantage is that it is applicable to a wide variety of CNNs. Figure 15 and Figure 16 show the results of GradCAM applied to the images in cancer_cells dataset. The red zone stands for the highlighting region that gives important features for predicting.

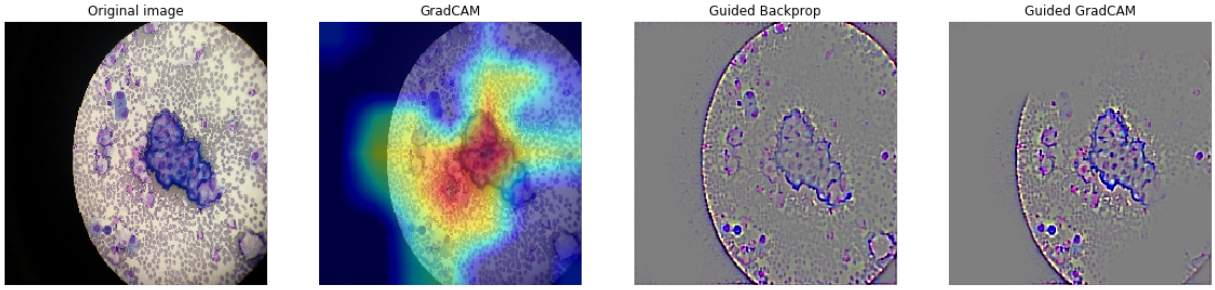

Figure 15: Cancer cell GradCAM representation on layer block5_pool

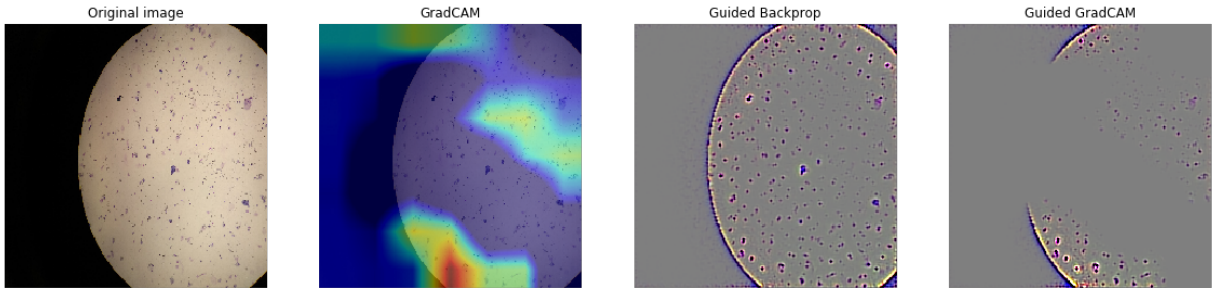

Figure 16: Non-cancer cell GradCAM representation on layer block5_pool

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

Table 5: Accuracy score using VLAD

4.2 PCA and VLAD

To improve the classification of images, some methods based on image vector representation are also often used. According to the Identity document classification as an image classification problem, features extracted from pre-trained CNN can be successfully transferred as image descriptors. Among different image descriptors, the more refined encoding results in the longer the global descriptor, and therefore the better performance.

Before applying these methods, Principal Component Analysis need to be employed to reduce the dimension of vectors. The number of dimensions is reduced to 128. Then we use VLAD as image descriptor, which is a simplified version of Fisher Vectors. It only takes the difference between descriptors and their closest centre of K-means clustering. Supposing a set of features $I = (x_1, x_2, .., x_n)$ extracted from an image, and μ_k are the cluster means which are the same dimension as features x_i. Features are encoded by VLAD by considering the differences between x_i and their closest centre μ_k:

$$v_k = \sum x_i - \mu_k$$

And the vectors are then stacked together as a global descriptor: $(v_1, v_2, ..., v_k)^T$. Its principle can be expressed as the following figure: An L2 normalization are also performed after VLAD. Because of the requirement of huge memory and storage when processing big data sets, we are only able to test on the relatively small data sets. We have also managed to reduce the size of descriptors of big data set by choosing them randomly, but the result was not very reliable. The result of small data sets is shown below:

We noticed that changing layers didn't have significant improvement on the accuracy score. The augmentation of number of k-means center also does not give better results on all data sets. It is evident that applying VLAD to the features from layer block5_pool have improved the performance of the classifier. However for layer fc1 and fc2, the results are not as good as before.

Layer	block5_pool	fc1	fc2
Accuracy(k=64)	0.82	0.71	0.82
Accuracy(k=128)	0.84	0.84	0.82

Table 6: Accuracy score using VLAD on miniMIT_Etus

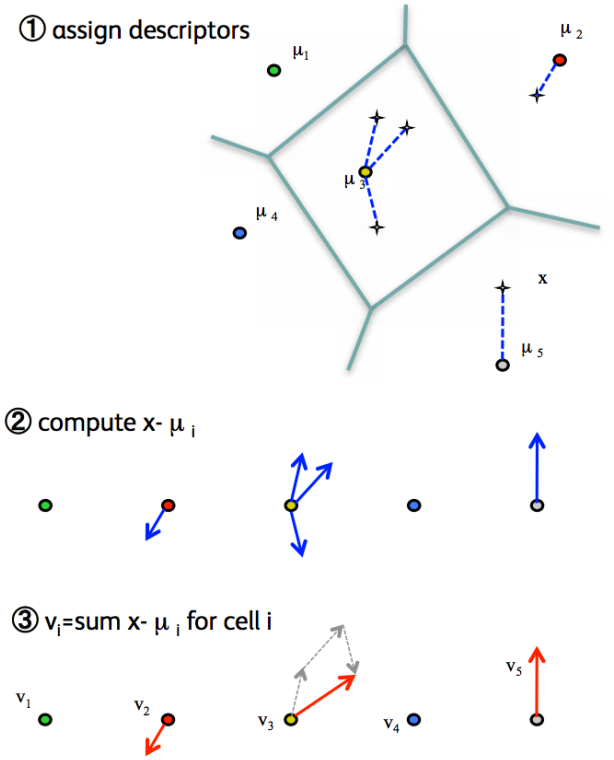

Figure 17: VLAD calculation process

Layer	block5_pool	fc1	fc2
Accuracy(k=64)	0.77	0.77	0.79
Accuracy(k=128)	0.74	0.71	0.74

Table 7: Accuracy score using VLAD on cancer_cells

4.3 Pre-trained models

The third strategy is to take a pretrained model - again the VGG19 - and try to find the best layers to fine-tune. Fine-tuning means that we will reload the training of our network on a chosen set of the original layers.

To start, we remove the last layer which is a dense layer with 1000 neurons (the number of classes for the imagenet dataset) and replace it by a dense layer with the number of classes of our dataset. The idea behind that is that such a network has been trained on a big amount of data (millions of images) and will be able to catch a lot of features useful for our task.

It gives us a nice baseline to start. Then, we try to go further and remove the weights of more layers : fc2, fc1 and then all the convolutional ones.

Results are presented on the tableau below for the Chest X-ray data set. Training was done for 300 epochs (we defined an early stopping callback at 10 epochs without any improvements). The optimizer chosen was Adam with a learning rate of 1e-6. We keep the model with the lower cross-entropy loss on validation set.

$$CrossEntropyLoss = -log\left(\frac{e^{sp}}{\sum_j^C e^{sj}}\right)$$

Results above are clearly better if the top layers are trainable and the bottom ones frozen. Training only the deep layers was predictable to give bad results. It shows also that we obtain our best results on the first epochs of the training. Thus we can predict our model to hugely overfit. A notable exception is the training where only the last layer was trainable. We got the best accuracy and a very good recall and our training history actually seems to learn something.

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

Table 8: Results of pre-trained VGG-19 with different trainable layers

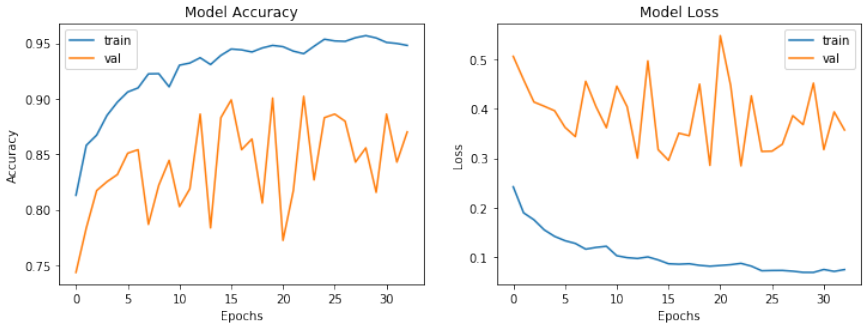

Figure 18: Training history when freezing blocks and fc1 layers

To check if our intuition is correct, we can visualize what our model is predicting. We will use two tools for that purpose, GradCAM and Shap - seems that CNN Fixations is also very good but we didn't try it.

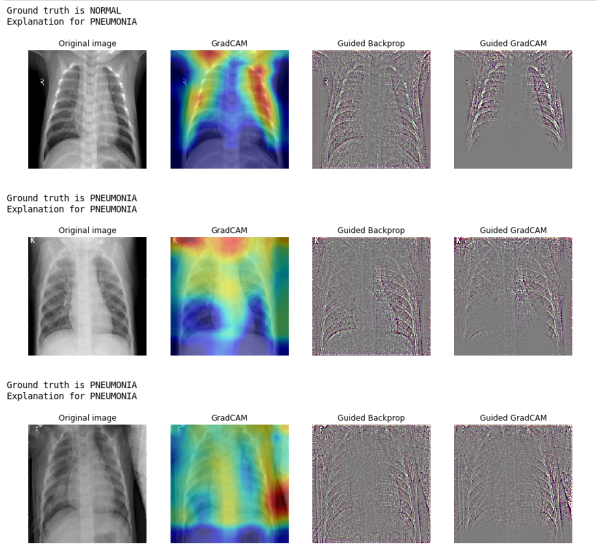

Figure 19: GradCAM of the block5_pool layer

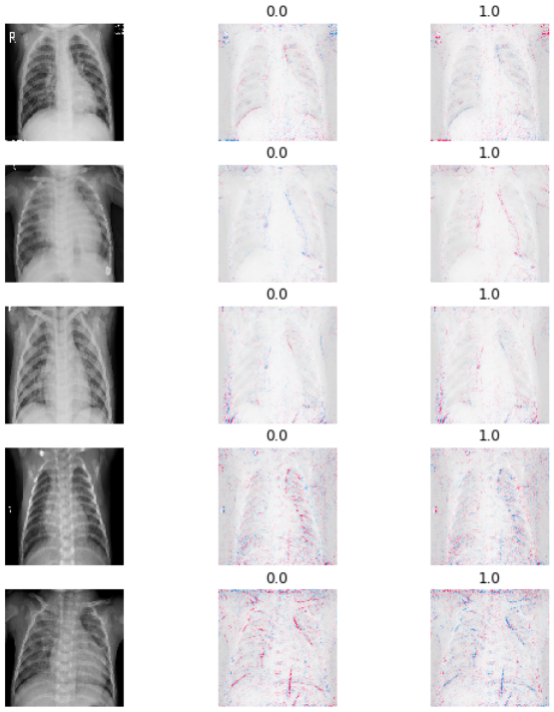

Figure 20: Shap DeepExplainer

Even if we have excellent results in terms of metrics (accuracy, precision, recall), we cannot be happy about what our model is predicting. This is mainly motivated by the observation that the features of the pre-trained model

contains features that should be useful for the tasks it was originally designed. **We seems to mainly predict *chest with pneumonia/normal* rather than *pneumonia/normal***.

For the sake of comparisons on other data sets, we got these results using the best method found earlier - only fc2 layer is trainable :

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

Table 9: Results of pre-trained VGG-19 on every data set

Up to that point, we can observe two major points :

- pre-trained models performs better on larger data sets.
- pre-trained models, even on larger data sets, are overfitted.

The first point was predictable and we asserted it on our introduction. The second point is a bit more tricky, we can see that the last two FC contains 4096 neurons each. In case the input image had shape (224, 224, 3), the last convolutional layer will have shape (7, 7, 512). So we got 7x7x512x4096 = 102.764.544 params to train only on the fc1 layer. Thus, since our data sets don't exceed 5000 train images, our model will certainly assign one neuron to each image... which is really really bad. We can move around that by adding one Dropout layer between each FC and decrease the number of neurons in our FC. We tried the following classification block. Sadly, we didn't get significantly improvements on metrics but the same results were found after a few more epochs. Hence, we can expect that the use of Dropout has decrease the impact of overfitting.

```
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
fc1 (Dense)                  (None, 1024)              102761472
_________________________________________________________________
dropout1 (Dropout)           (None, 1024)              0
_________________________________________________________________
fc2 (Dense)                  (None, 512)               524800
_________________________________________________________________
dropout2 (Dropout)           (None, 512)               0
_________________________________________________________________
predictions (Dense)          (None, 2)                 1026
=================================================================
```

Figure 21: Classification block with Dropout/Dense layers

We try the same experiments on other data sets and found that the only interesting GradCAM we obtained on the mini MIT data set as shown on next figure.

These results are really interesting : since the 'bookshop', 'library' and 'bus' categories exist in the original data set, **our model for that data set can be qualified as *strong* because it can motivates its output**. On another side, the other data sets we explored were really different of the original data set. Hence the weird outputs even with excellent metric results. **Thus, we decide to explore a way to improve our network in terms of interpretation**.

4.4 Fine-tuning

The strategy here is not only replace and retrain the classification block on top of the ConvNet, but also fine-tune the weights of the pre-trained network. As we already say, the data sets used are very different from the original data

Figure 22: GradCAM of the block5_pool layer

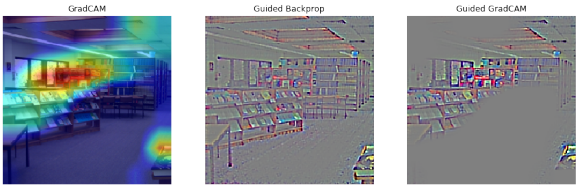

Figure 23: Another GradCAM of the block5_pool layer

set, then we can predict our training to not get profit from the weights of the full network. **It's mainly motivated by the observation that the earlier features of a ConvNet contains more generic features that could be useful for many tasks, but the late layers becomes progressively more specific to the details of the classes contained in the original data set**. Thanks to Class Activation Mapping algorithm again, we can check which convolutional layers are useful for our task.

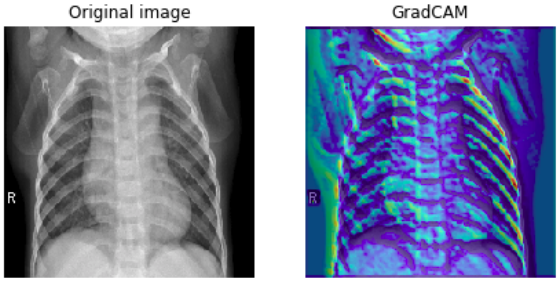

Figure 24: Block1_pool

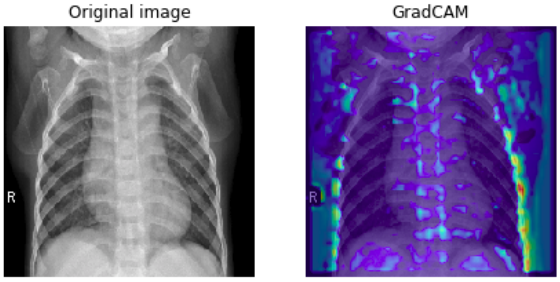

Figure 25: Block2_pool

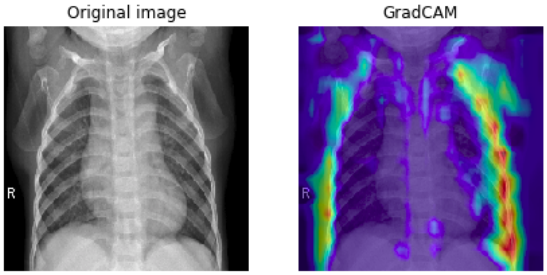

Figure 26: Block3_pool

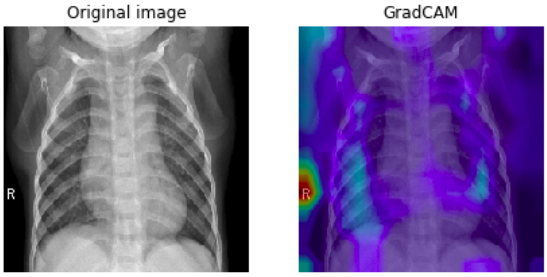

Figure 27: Block4_pool

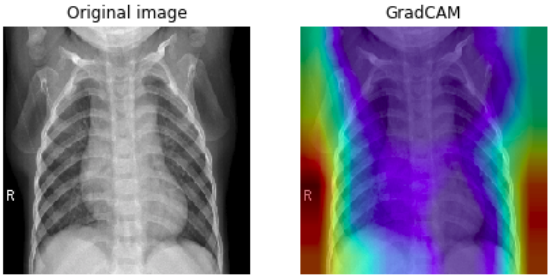

Figure 28: Block5_pool

We see that the first three layers can find useful information for our specific task. Thus we will find a way to train our specific layers on top of such layers.

4.4.1 Xception network and Depthwise Separable Convolution

The Xception network proposed by François Chollet is detailed on the next figure.

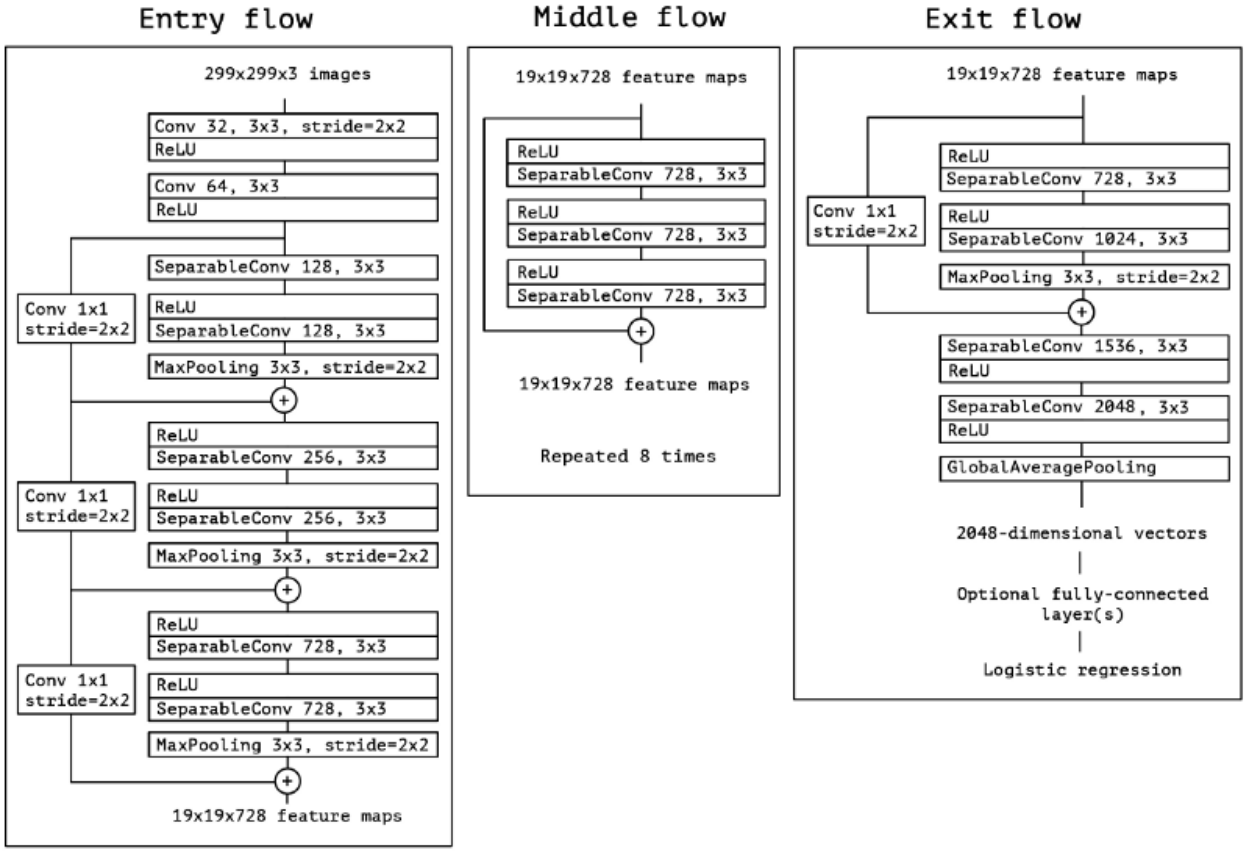

Figure 29: Xception overview

That network is build on top of SeparableConv which are Depthwise Separable Convolutions.

Let's look at an example to understand the differences between a normal convolution and a depthwise separable convolution.

Assert we have a kernel of size 5x5 with 3 channels (the standard to represent 3D objects or to use the RGB standard) applied on a 12 x 12 image, we would have Z 5 x 5 x 3 kernels that moves (12-5) x (12-5) times. Thus it does 4800 x Z multiplications.

In the case of separable convolution, since we separate in 2 steps, we will have 3 kernels of shape 5x5x1, each moving 8x8 times. That gives us 4800 multiplications. In the second step, the pointwise convolution, we have Z 1x1x3 kernels that move 8x8 times, leading to 1 x 1 x 3 x 8 x 8 = 192 Z multiplications.

We end up with two formulas for an input image of 12 x 12 x 3 pixels and a 5 x 5 x 3 convolution. Z denotes the number of channels, usually 256 for RGB standard.

Normal convolution : 4800 Z multiplications Depthwise convolution : 4800 + 192 Z multiplications

You can see difference like this : **in the normal convolution, we transform the image as many times as we got channels. On the opposite, in a separable convolution, we transform the image once (depthwise convolution) then we elongate it to the number of channels desired (pointwise convolution)**.

Getting back to our neural network, using depthwise convolutions drastically reduces the number of parameters for a block of convolution but it doesn't reduce significantly performance according to François Chollet paper. His idea was to add a lot more layers with fewer parameters. He obtained better results than VGG-19 on ImageNet data set.

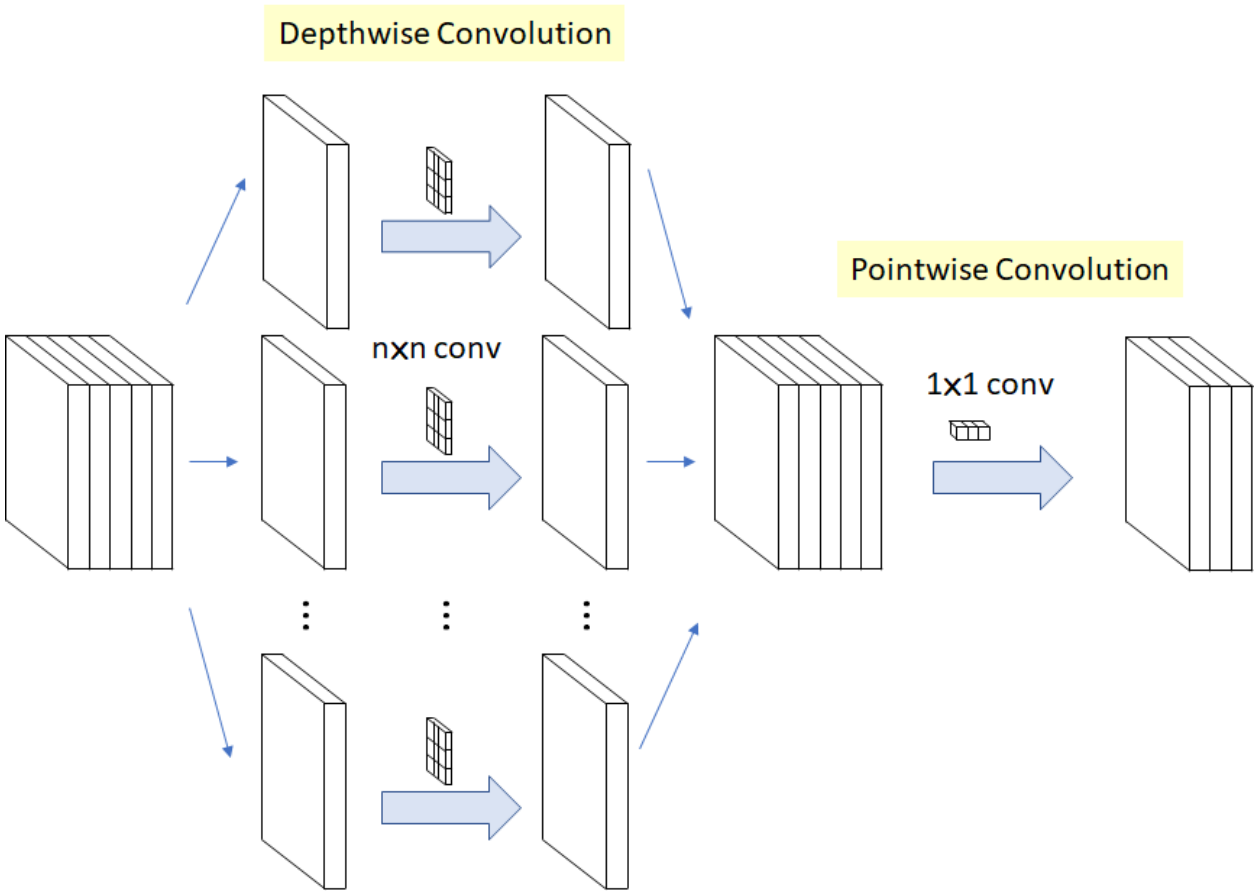

Figure 30: Depthwise convolution

4.4.2 Our custom convolutional neural network built on top of VGG-19 and Depthwise Convolutions

It would be interesting to explore that network exactly as we did with VGG-19, and keep the architecture with the exception of the last fully connected layers replaced in the same way we explained earlier.

Instead, we explore a different approach, keeping in mind that we want to get a better explanation of our predictions and try to combine the VGG-19 architecture with Xception by replacing the VGG-19 Conv2D layers to SeparableConv2D followed by BatchNormalization in order to speed up the training part (remember, we'll have less parameters). We use the following architecture for our network.

The VGG-19 part is initialized with the weights of the pre-trained network and we make them not trainable. All the other layers are randomly initialized. We obtained way better results.

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

Table 10: Results of our custom CNN

input_3 (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

Figure 31: Block_1 to 3 from the VGG-19 network

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

Figure 32: We add two blocks of SepConv2D combined with BatchNormalization

4.5 Histogram equalization - CLAHE

We try to use the Contrast Limited Adaptive Histogram Equalization for enhancing the local contract of images and see if we can get better results. Even if the images are way more readable for humans, we didn't get better results. An interesting feature that we didn't compute would be to combine both images to see if we can improve our predictions.

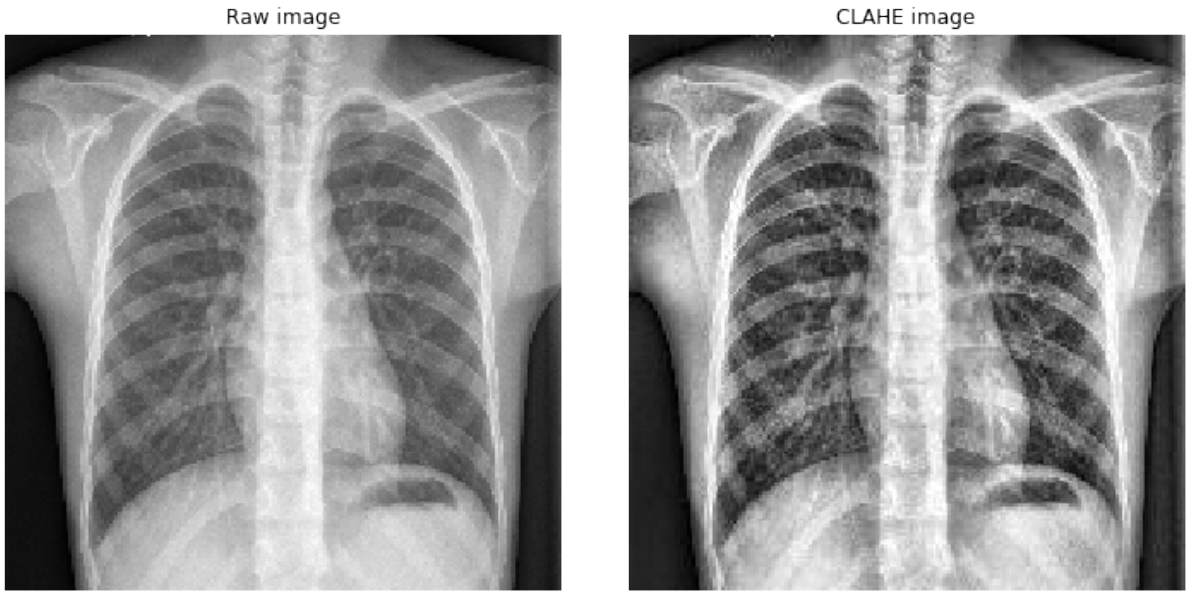

Figure 33: Difference between a raw and normalized image from Chest X-ray data set

4.6 Data Augmentation

We lately tried data augmentation for the small datasets in order to see the impact of increasing data. We tried Albumentations and Imgaug. Data augmentation prevents the network to see the same image twice during the training process.

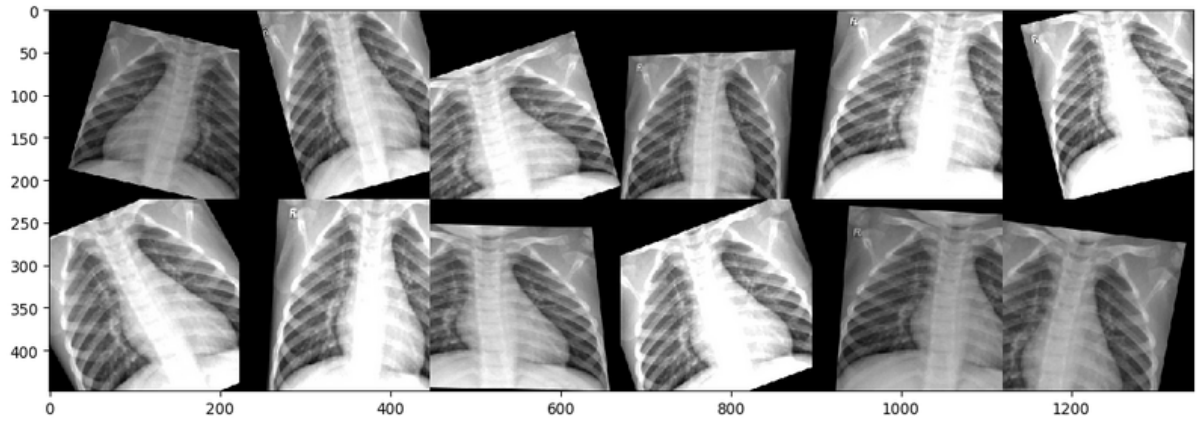

Figure 34: 16 differents views of a same image from the Chest X-ray data set

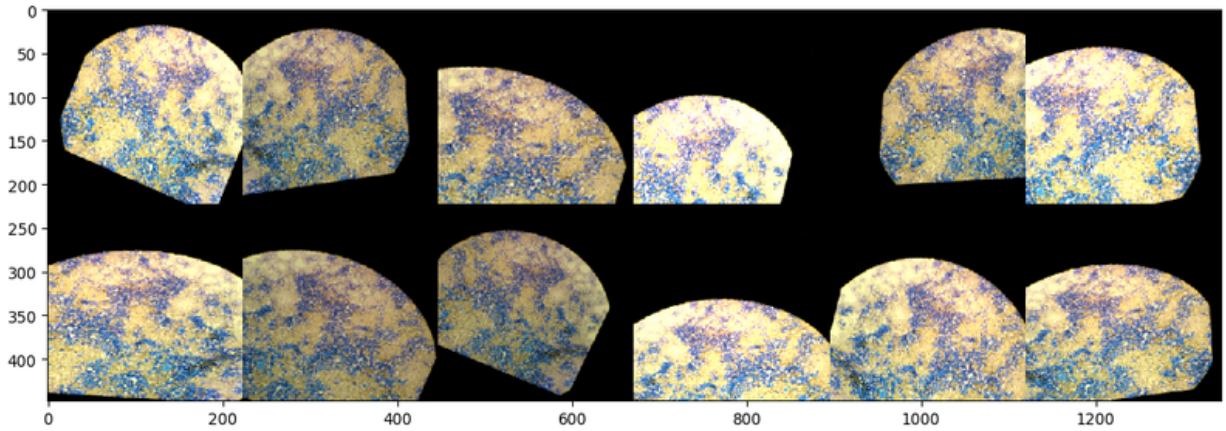

Figure 35: 16 differents views of a same image from the Cancer Cells data set

Data augmentation offered excellent improvements.

Data set	Accuracy with augmentation	Accuracy without augmentation
Chest X-ray	0.92	0.902
Kvasir (version 2)	0.89	0.82
Mini MIT	0.83	0.77
Cancer cells	0.82	0.78

Table 11: Results with data augmentation

5 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, -

Table 12: Final results

6 Conclusions and future improvements

When, how and which Transfer Learning scenarios should you use for your next image classification project ?

With our experiments on 4 data sets of various properties, we can conclude that using one or another type of transfer learning you should perform depends of several factors. The most important ones are the size of the data set and its similarity to the original data set.

If the data set you use is quite large (at least thousands of images), it can be a good move to try to fine-tune through the full network. In case the data is similar to the original data set, you can probably be enough confident to keep the same architecture the original network had. If the data is very different from the original data set (as in Chest X-ray Pneumonia), it can be a good idea to explore a network and keep weights and architecture of the first layers, which are very generic, then train again the network by running backpropagation.

If the data set you use is small, and in our case it was *really* small with under 250 images, it might not be a good idea to fine-tune the whole network due to overfitting concerns. Data augmentation can be a way to lower the overfitting and actually learn something.

In case the data is similar to the original data (as in mini MIT Etus), the best idea might be to train a linear classifier (SVM, PCA, VLAD) on the fully connected layers extracted directly from the network. In general, extracting features from a fully connected layer with a linear SVM classifier would have better result than features from block5_pool. Applying VLAD to the features extracted from block5_pool would improve the accuracy, however it's not the case for features from layer fc1 and fc2. Also, keep in mind that if you want to do recognise objects in ImageNet data set, it would be better to simply use one of the best neural network pre-trained for weeks on multiple GPUs. For example, the Xception network obtains 0.99 accuracy on the mini MIT Etus data set.

Also, we found that even an excellent neural network in regards of metrics can be qualified as weak because it cannot motivate his prediction. Our custom neural network outperforms the accuracy state-of-art on Chest-Xray test set but it is far from usable in real world problem because it's way overfitted and not usable by radiologists. This is way we strongly encourage to always check what a neural network predict and give more attention on interpretation than pure metrics.

References

[1] Ronan Sicre, Ahmad Montaser Awal and Teddy Furon Identity Documents Classification as an Image Classification Problem *Battiato S., Gallo G., Schettini R., Stanco F. (eds) Image Analysis and Processing - ICIAP 2017. ICIAP*, 2017.

[2] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, and Jianming Liang Convolutional Neural Networks for Medical ImageAnalysis: Full Training or Fine Tuning? *arXiv preprint arXiv:1706.00712*, 2017.

[3] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, Cordelia Schmid Aggregating local image descriptors into compact codes *IEEE Transactions on Pattern Analysis and Machine Intelligence, Institute of Electrical and Electronics Engineers, 2012, 34 (9), pp.1704-1716. <10.1109/TPAMI.2011.235>. <inria-00633013>*

[4] François Chollet, Google Inc. Xception: Deep Learning with Depthwise Separable Convolutions *arXiv preprint arXiv:1610.02357*, 2017

[5] Maciej A. Mazurowski, Mateusz Buda, Ashirbani Saha, Mustafa R. Bashir Deep learning in radiology: an overview of the concepts and a survey of the state of the art *arXiv preprint arXiv:1802.08717*, 2018

[6] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, Andrew Y. Ng CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning *arXiv preprint arXiv:1711.05225*, 2017