Augmenting Convolutional Networks with Attention-based Aggregation for Image Classification and Breast Cancer Detection

Filippo Betello*
Sapienza University of Rome
Neural Networks
Rome, Italy
betello.1835108@studenti.uniroma1.it

Federico Carmignani*
Sapienza University of Rome
Neural Networks
Rome, Italy
carmignani.1845479@studenti.uniroma1.it

***Abstract*—This paper shows how to augment any convolutional network with an attention-based global map to achieve a global reasoning. The usual average pooling is substituted by an attention-based aggregation layer, inspired to a single transformer block, with the goal of weighting the patches involved in the classification decision. This layer is connected to a patch-based convolutional network characterized by 2 parameters: width and depth. The architecture keeps the resolution of the input unchanged across all the layers and it helps to reach an optimal balance between accuracy and complexity. We show how this deep learning network can be applied to a common dataset, like CIFAR-10, with suprising results. Then our goal is to show the application of this method on a popular task in Artificial Intelligence and Machine Learning: Breast Cancer Detection, where the attention maps (heatmaps) produced point out the areas having a greater probability to be affected by Cancer.**

I. Introduction

A. CNN

Convolutional neural networks (CNNs)[**1**] were designed for image recognition tasks and were originally applied to the challenge of handwritten digit recognition (Fukushima 1980; LeCun 1989). The basic design goal of CNNs was to create a network where the neurons in the early layer of the network would extract local visual features, and neurons in later layers would combine these features to form high-order features. A typical CNN consists of two parts: first, we have a few *Convolutional Layers*, whose job is to extract the features of the input image and bring them into a form that is easier to process without losing the critical features. This information is then passed through few *Fully Connected Layers*, whose job is to classify the received information into a category. A single *Convolutional Layer*, as shown in Figure 1, consists of a set of filters or kernels, matrices that contain the learnable weights (the weights are what gets adjusted when the feature extractor learns). These filters are much smaller than the input image, so they propagate over the whole image. The results of each step are then put together to form a new image. This output image is then used as input to the next ConvLayer. Depending on the filter weights, different aspects of the image are enhanced. Two types of pooling operations often used are called *average-pooling* and *max-pooling*. Average-pooling outputs the average pixel-activation of the current set of pixels it is applied to. The max-pooling operation outputs the maximum pixel-activation out of the patch it is applied to. These layers help to reduce the size of the input, as well as the amount of redundant information.

*equal contribution

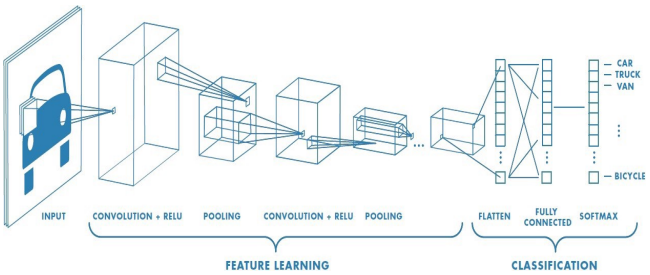

Fig. 1. Convolutional layer

II. ResNet

We will not enter into the details of the model[**2**], but we describe the general idea. According to the universal approximation theorem[**3**], we know that a feedforward network with a single layer is sufficient to represent any function. However, the layer might be massive and the network is prone to overfitting. Deep networks are hard to train because of the vanishing gradient problem, as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly. The authors addressed this problem by introducing deep residual learning framework so for this they introduce skip connections that perform identity mappings, as shown in Figure 3.

ResNet-50 is a convolutional neural network that is 50 layers deep (48 Convolution layers along with 1 MaxPool and 1 Average Pool layer). We use the ResNet-50 architecture

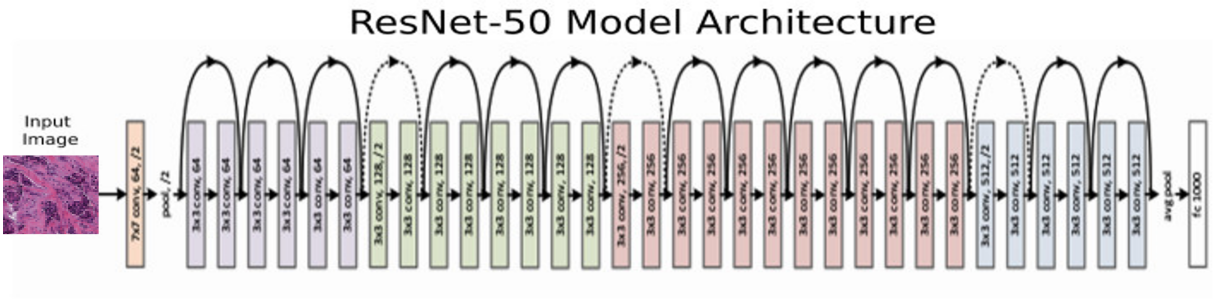

Fig. 2. ResNet-50 architecture. In input there is an image of a breast affected by cancer.

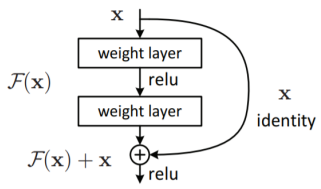

Fig. 3. Skip connection

as a basis for comparison with the innovative architecture PatchConvNet described in the following section.

III. PatchConvNet

PatchConvNet[**4**] is a deep neural network created merging features of two popular models: Vision Transformers and Convolutional networks.
ViT has several characteristics:

- patch based processing;
- aggregation of the image information based on a so called "class token", used in comparison with the patches more related to the classifcation decision;
- a softmax in a self-attention block for producing attention maps: they show relationship between class token and patches.

Maps are created for visualization purposes[**5**; **6**] but they have loose interpretability: many softmax and heads can be used.

ConvNets are convolutional networks and our PatchConvNet is:

- a convolutional network;
- using patch processing;
- producing attention maps with a single class token.

The usual average pooling layer is substituted by a learned attention-based pooling layer: a simple way to weigth the patches, multiple layers and heads in aggregation with a single weight per patch, a single contribution in the weighted sum summarizing the images. Attention maps are extracted from ViT using a visualization procedure[**5**]. ResNet and PatchConvNet are used augmented with the attention based aggregation layer.

ResNet has a hierarchical model in order to reduce resolution increasing working dimensionality but at the same time it has a low resolution maps: a balance is needed between low complexity and optimal maps.

A. History

The attention based processing has been introduced by ViT[**6**]: it processes images as a set of non-overlapping patches without convolutions or downsamplings. Many works are now reintroducing them in ViT like Swin[**7**] in order to reduce the spatial features resolution improving computational efficiency (FLOPs) and memory usage but reducing at the same time the quality of their attention maps. The use of patches[**8**] have been proposed beyond Transformers in MLP based networks showing the potentiality of patch-wise convolutions.

The classification decision is obtained through a self-attention mechanism offering a direct access to the location of the information used to make it, inspired to the idea of Vision Transformers.

B. The architecture

The goal of this innovative architecture is to combine a high resolution feature map with the mantainance of the resolution all across the layers. The first point can be achieved also using a regular ResNet-50, whereas PatchConvNet allows to accomplish both the objectives: we have done experiments using both the networks for comparison purposes. The PatchConvNet architecture is shown in Figure 4 and it is characterized by the embedding dimension d and the number of repeated blocks in the trunk N.

The convolutional stem is the first part of the network where the images are segmented and mapped into a set of vectors: the patches. This operation is done adopting a convolutional procedure for stability reasons: four 3×3 convolutions followed by a GELU non-linearity. This is applied to images of size $W \times H \times 3$ and produces a vector map according to $W/16 \times H/16 \times d$ in case of patches whose size is 16×16 or $W/4 \times H/4 \times d$ in case of patches whose size is 4×4. It can be seen as a set of k d-dimensional patches.

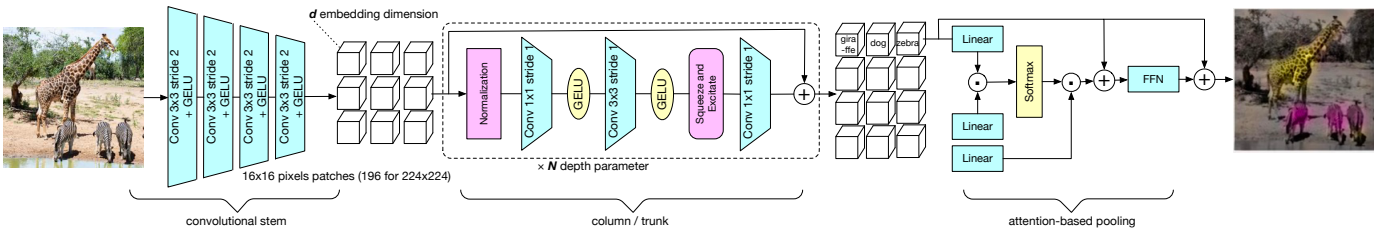

Fig. 4. The full architecture, with the convolutional stem on the left, the convolutional main block in the middle and the attention-based pooling on the right.

The column is the core of the network and it consists of N residual convolutional blocks, each one is characterized by a normalization, a 1×1 convolution, a 3×3 convolution and a squeeze-and-exciting layer for mixing channel-wise features. Finally a 1×1 convolution right before the residual connection is present. The first two convolutions are followed by a GELU. The output of this block has the same shape of the input: the number of vectors representing the patches. The normalization used is LayerNorm since the use of BatchNorm is less optimal when models and input are large since it depends on the batch size.

Attention-based pooling is at the output of the trunk where the pre-processed vectors are aggregated using a cross-attention layer. A query class token aggregates them as a weighted summation. The weights depend on the similarity of projected patches with a classification token (CLS) similar to a class token. The resulting d-dimensional vector is added to the CLS and then passed to a feed-forward network (FFN). It uses only a single block and a single head avoiding the diluition of attention across multiple channels. Then a single softmax is used reflecting how the patches are weighted.

C. Details

The model is simply parametrized by width and depth. The resolution used by us is 384 and we refer by S the model with a vector size of $d = 384$ per patch. The $S60$ model is chosen since it has similar number of parameters and complexity as a ResNet-50. The model allows to keep a constant resolution across the trunk. The resolution can be easily changed, therefore this network is quite flexible, there is no max-pooling or other non-reversible operator in our architecture. We replied the paper results also trying to make some ablations in order to find new ways of expansion of this architecture: the dataset we used is CIFAR-10 for general image classification and then we applied this method at Breast Cancer Detection. Therefore we tried 2 approaches:

- patches 4×4 in images 32×32;
- patches 16×16 in images 32×32.

IV. Preparation for training

This section highlights the differences between our choices and the one of the reference paper[**4**]. We use a different dataset, number of epochs and loss function. Furthermore, we use the same optimizer and we include an additional study using different optimizers and learning rates.

A. Dataset

The dataset used is the popular CIFAR-10: it consists of 60,000 colour images, whose size is 32x32, divided in 10 classes (airplane, car, bird, cat, deer, dog, frog, horse, ship, truck), with 6,000 images per class. We use the usual split of this dataset, composed of 50,000 training images and 10,000 test images. In the reference paper, they use ImageNet: it consists of over $14x10^6$ images belonging to more than 20,000 categories. We opt for CIFAR-10 due to its smaller dimension.

Data augmentation. We decide to apply some data augmentation functions to try to reduce the overfitting. We apply these functions only to the training set. In particular:

- We flip the images with respect to the horizontal axis (*RandomHorizontalFlip*);
- We rotate the images to a specified fixed angle of 10° (*RandomRotation*);
- We performs actions like zooms, change shear angles (*RandomAffine*);
- In the end we randomly change the brightness, saturation, and other properties of an image (*ColorJitter*).

We use the same batch size of the paper (2048) for two main reasons: first of all running the code in Google Colab requires a lot of time, as we will see in details in the next section. In addition to that, Google Colab limits the memory up to 12GB and there are time limitation for using the virtual GPU resource.

B. Optimizer

We adopt the same optimizer of the paper: Lamb[**9**], a variant of AdamW[**10**]. The principle is the layerwise adaptation strategy, used in order to accelerate training of deep neural networks using large mini-batches. Using this approach, they develop a new layerwise adaptive large batch optimization technique called Lamb. We make also preliminary experiments with other optimizers, in particular SGD[**11**] and Adam[**12**], which results are reported in the next section.
For the loss function, we use the CrossEntropyLoss that is useful when training a classification problem with *C* (in our case C=10) classes.

C. Training

We train our models during 50 epochs using a *learning rate* (lr) of $3x10^{-3}$ or 0.01. For both of this values we fixed the *weight decay* (wd) to 0.01, as in the paper. In addition to that, we insert also the *Early Stopping*: it is a form of regularization used to avoid overfitting on the training dataset. Early stopping keeps track of the validation loss, if the loss stops decreasing for 10 epochs, the value for the patience, in a row the training stops. It will save a checkpoint of the model each time the validation loss decrease.
In order to compare the models, we used the accuracy and computational time to achieve such result. In the next section results are reported.

V. Experimental results

In this paragraph we present our main experimental results in Image classification based on the CIFAR-10 dataset, using the ResNet-50 architecture and the S60 PatchConvNet architecture, in contrast with the one presented in the reference paper[**4**].

A. Their classification results

As written above, they used the ImageNet dataset training for 400 epochs with a batch size of 2048 and a learning rate fixed at $3x10^{-3}$. In addition to that, they fixed the weight decay to 0.01. Their results shows that ResNet-50 achieve an accuracy of **80.4%**, while PatchConvNet S60 model obtain an accuracy of **82.1%**. Moreover, they add more evaluation metrics, measured for a constant batch-size of 256 images:

- Throughput (im/s): in ResNet-50 is equal to 2587, while in S60 is equal to 1125;
- Peak memory (MB): 2182 for ResNet-50, 1321 for S60;
- FLOPs ($x10^9$): in ResNet-50 are 4.1, on the other hand in S60 are 4.0;
- Number of parameters ($x10^6$): 25.6 for ResNet-50, 25.2 for S60.

In consideration of the above results, S60 moderately exceedes the performance of ResNet-50, but is more useful thanks to the heatmaps that it can produce.

The authors also presented some modification to the architecture:

- Class-attention $\rightarrow$ average pooling: average pooling is the most common aggregation strategy in ConvNet while class attention is only used with ViT. This modification bring the accuracy down to 81.9%;
- Conv-steam vs linear projection for the patch extraction in the image: the result accuracy is 80%, showing that the convolutional steam is the key for best performance;
- Layer-normalization $\rightarrow$ batch-normalization: improves the performance a bit (82.4%);
- Single-head attention vs multi-head attention: performance is not affected so much (81.9%), while the first choice reduce memory consumption and simplifies attention map visualization.

B. Our classification results

We use the CIFAR-10 dataset with same batch size of 2048 for 50 epochs, using ResNet-50 and S60 architectures. We present a comparison between two optimizers, SGD and Lamb, and two values for lr that are 0.01 and $3x10^{-3}$. Table 1 shows the main results both with SGD and with Lamb. Our code, as their, depend on the PyTorch[**13**] and timm libraries[**14**].

TABLE I
Results for CIFAR10 with 4 patches

	Accuracy			
	SGD		**Lamb**	
lr	0.01	$3x10^{-3}$	0.01	$3x10^{-3}$
ResNet-50	67.39	56.51	77.4	75.36
PatchConvNet S60	59.72	55.82	**78.2**	68

The table shows that the PatchConvNet S60 architecture is better than ResNet-50, like in the reference paper, with maximum accuracy reached at **78.2%** (in green in Table 1). The computational time is linked to the Google Colab computation capacity: on average the S60 architecture needs 2:20 hours for training, while ResNet-50 needs about 1 hour and 15 minutes.
However, there is a limitation on these results: we extract only 4 patches, with the dimension of 16x16, to build the attention map. On the other hand, in the paper there are 196 patches of the dimension of 16x16 because the images are 224x224 (not 32x32).

C. Results with modification in the architecture

The previous subsection point out the fact that we build only 4 patches (16x16), because our images are 32x32. In order to deal with this problem we modify the dimension of the patch, decreasing their dimension to 4x4. Doing that we obtain 64 patches.
Then we perform more training: we changed the batch size to 64 and fixed the lr to $3x10^{-3}$. Some early experiments identify that the value of 0.01 for the learning rate is not good in this instance. In addition to that, the weight decay is fixed to the value of 0.01. We train for 10 epochs, because of time limitation on Google Colab.
In Table 2 we report the results both with SGD and with Lamb.

TABLE II
Results with CIFAR with 64 patches

	Accuracy	
	SGD	**Lamb**
ResNet-50	68.76	75.32
PatchConvNet S60	51.10	**81.54**

As expected, the results are much better than the previous one, considering also the fact that the training lasts only on 10 epochs. The motivation is that there are more patches, so the classification is more accurate. On the other hand, the computational time is negatively affected: the average duration

for the S60 model is 2:40h, but we are dealing with limited resource.

D. Attention Maps

In addition to the better accuracy results than ResNet-50, PatchConvNet architecture is useful to build the attention map: we compare our maps, based on the architecture with patches 4x4 that obtained the 81.54% accuracy, with the one of the paper with patches 16x16 and an accuracy of 82.1%.
In Fig. 4 is possible to see this comparison:

- On the left there is the original image, taken from the dataset;
- In the middle there is our results;
- On the right there is their results.

We use four different images: duck, plane, car and a frog.

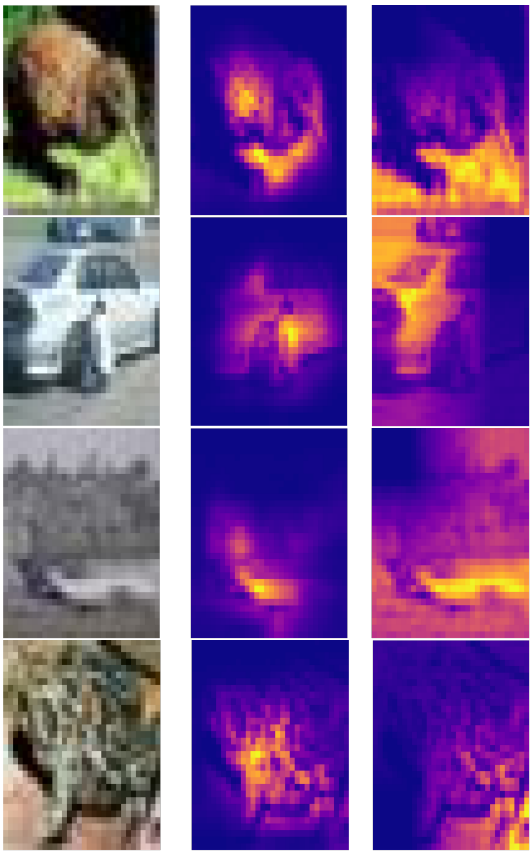

Fig. 5. Attention Map on a duck, car, plane and a frog. In the middle there is the map based on our result.

The results are outstanding: our model perform as good as their. Actually our maps seems more accurate: in our attention maps only the most important things in the image are highlighted, while their attention maps is more general highlighting also irrelevant pieces.
In addition to that, we didn't perform any parameter tuning and we train for only 10 epochs. So it is possible to have even better results, using transfer learning or training for more epochs.

VI. Application to Breast Cancer Detection

Breast cancer is the second leading cancer-related cause of death among women in the US. Every year in Italy 50,000 new cases are diagnosed among women and 500 among men. Between them, 87% survive after the 5 years from the diagnosis of the breast cancer and only 80% after 10 years. There is the necessity of large screening, but there has been discussion regarding the potential harms of screening, including false positive recall and associated false positive biopsies. In fact, the majority of women undergo another mammogram and/or ultrasound for clarification. Among them only 10-20% are recommended to undergo a needle biopsy for further work-up. Among these, only 20-40% yield to a diagnosis of cancer. Evidently, there is a unmet need to shift the balance of routine breast cancer screening towards more benefit and less harm.
Attention maps can be a solution to that: they can highlight the patches of the image that brought the model to the classification of a cancer.

A. Training receipts

Dataset used is taken from Kaggle[**15**]. Originally there were 277,524 patches of size 50 x 50 were extracted, 198,738 IDC (Invasive Ductal Carcinoma, the most common subtype of all breast cancers) negative and 78,786 IDC positive. We take a subset of this pulling out 52,521 patches, 38,618 IDC negative and 13,903 IDC positive, keeping more or less the same ratio between positive and total patches (28% vs 26%). Then we split the images into training set (80%) and test set (20%).

Data Augmentation is applied both to the training and the test set. In particular:

- We flip the images with respect to the horizontal axis (*RandomHorizontalFlip*);
- We flip the images with respect to the vertical axis (*RandomVerticalFlip*);
- We rotate the images to a specified fixed angle of 10° (*RandomRotation*);

We decide to keep the original colors because otherwise it could be more difficult to classify the images.

B. Training

We used the same parameters as in the CIFAR-10 with 64 patches (lr=$3x10^{-3}$ and wd=0.01) and the same two different optimizers (SGD and Lamb). In addition to that, we train for 15 epochs with a fixed batch size of 64.
The results are described in Table 3.

The best result is obtained with the ResNet-50 architecture, but on average the S60 model performs better. Moreover, the computational time is similar as before: 2:30h for S60 model and 45 minutes for ResNet-50. Later on we discuss an ablation on the hyper-parameters in order to find the best combination to achieve better results and understand if it is a robust method for breast cancer classification.

TABLE III
RESULTS WITH BREAST CANCER DATASET

	Accuracy	
	SGD	**Lamb**
ResNet-50	62.19	**85.39**
PatchConvNet	78.02	84.94

C. Attention map

In figure 5 is possible to see the attention maps build with three different images of positive invasive ductal carcinoma.

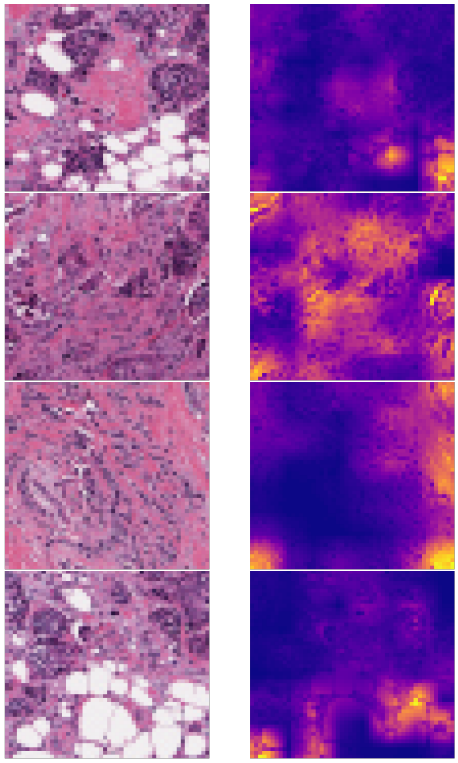

Fig. 6. Attention Map on breast cancer images.

The results shown with the CIFAR-10 dataset were easier to read: everyone can point out where in the figure are the most important things of an image. Here is not obvious as before, because we don't know which patch is carcinogenic or not. It could be useful to compare the results with more attention maps obtained with same architecture but different training accuracy.

D. Ablations

We perform more training by changing the learning rate to three different values (0.01, 0.0075 and $3x10^{-3}$) and by varying the weight decay from 0.01 to 0, but keeping the batch size to the fixed value of 64. In addition to that, we also try three different optimizers (SGD, Lamb and Adam). The results of this ablation are reported in Table 4.
This exploration phase is being done only for the PatchConvNet S60 model, in order to verify if this model is robust enough in multiple condition.

TABLE IV
ABLATION RESULTS

		Accuracy		
lr	*wd*	**SGD**	**Lamb**	**Adam**
0.01	0	**85.42**	78	73.5
	0.01	78.2	78.74	73.24
0.0075	0	82.42	77	73.56
	0.01	77.63	80.76	73.56
0.003	0	80.55	81.82	72.68
	0.01	78	83.6	78.48

The table shows that both SGD and LAMB are robust optimizer, for all values of lr and wd. Moreover, the accuracy reached with lr=0.01 and wd=0 with SGD as optimizers prevails over the accuracy reached by ResNet-50 in the table 3. On the other hand, Adam confirms being a bad optimizers in this case.

In addition to that, we perform also fine tuning: using the best pretrained S60 model in Table 3, we train for 15 epochs using the same parameters. The accuracy reached is **87.95**%: as expected it is much better than before. Then we build the attention maps with this model, with the same images used before. The results are shown in Figure 7.

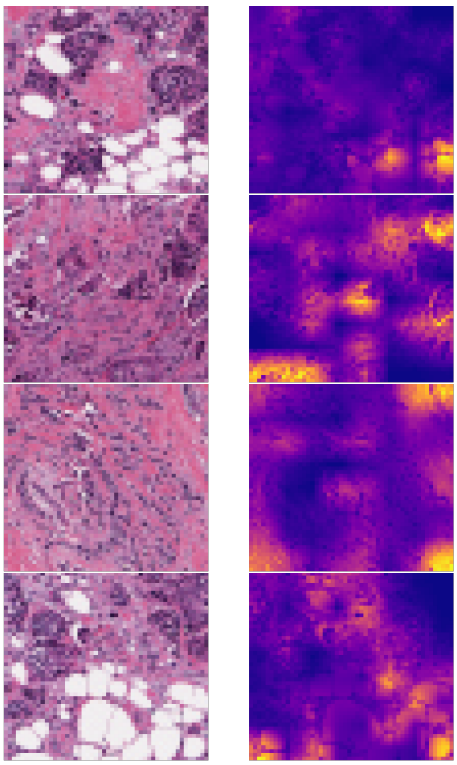

Fig. 7. Attention Map on breast cancer images with fine tuning model.

VII. CONLUSION

In this paper, we introduced a full patch-based ConvNet with no pyramidal structure. We used an attention-based pooling on top of the trunk, akin to the attention mechanism

in transformers, which offer visualization properties. We demonstrated its better result in classification common images with respect to the popular ResNet-50 architecture. Furthermore, the attention maps based on our predictions seems more accurate than their. Then we performed a classification task based on breast cancer dataset: the results were good and we proved that it is a robust method with the ablations.

Extensions: PatchConvNet can be extended in several aspects and modified in order to be applied to other tasks:

- Multiclass PatchConvNet: one token for each class and not one single token, an attention map for each class is produced: more accuracy but even more complexity, so it should be used with a lower learning rate and smaller batch size.
- Segmentation with PatchConvNet: it can be applied to semantic segmentation like in the paper is presented.
- Detection with PatchConvNet: it can be used for image detection and instance segmentation observing how it is on par with state of the art architectures.

We believe that there are some possibilities to achieve better results: firstly training for much more epochs without power limitation. Then another choice could be performing much more ablation studies in order to find the best values as learning rate and weight decay, according to the best optimizer chosen.

REFERENCES

[1] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Accessed: 2022-02-24.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition, 2015.

[3] Kurt Hornik, Multilayer Feedforward Networks are Universal Approximators, 1989.

[4] Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Piotr Bojanowski, Armand Joulin, Gabriel Synnaeve, Hervé Jégou. Augmenting Convolutional networks with attention-based aggregation, 2021.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. arXiv preprint arXiv:2104.14294, 2021.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021.

[7] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030, 2021.

[8] Ruiyang Liu, Yinghui Li, Dun Liang, Linmi Tao, Shi-Min Hu, and Hai-Tao Zheng. Are we ready for a new paradigm shift? a survey on visual deep mlp, 2021.

[9] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In In-ternational Conference on Learning Representations, 2020.

[10] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. arXiv preprint arXiv:1711.05101, 2017.

[11] Sebastian Ruder, An overview of gradient descent optimization algorithms, 2017. https://arxiv.org/pdf/1609.04747.pdf.

[12] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[13] Pytorch. https://pytorch.org/vision/stable/index.html. Accessed: 2022-02-08.

[14] Ross Wightman, Timm library. https://github.com/rwightman/pytorch-image-models. Accessed: 2022-02-08.

[15] Breast cancer dataset. https://www.kaggle.com/paultimothymooney/breast-histopathology-images. Accessed: 2022-02-25.