Histopathologic Cancer Detection on PCam

Stanford CS231n Final Report

Paul Caron
Stanford University
pcaron@stanford.edu

Julien Veron Vialard
Stanford University
jvialard@stanford.edu

Abstract

In this project, we study the possibility to apply Convolutional Neural Networks for Computer Vision to tissue analysis for early stage cancer detection. It is inspired by the contest of PCam dataset [8]. Using different CNN architectures, we create various binary classifiers and evaluate their quality with Receiver Operating Characteristic curves. Using deep residual neural networks and ensemble learning techniques, we aim at competing state-of-the-art classifiers for Histopathologic Cancer Detection. Finally, we analyze our results using specific metrics for medical purpose. The goal is to optimize our model for real-life applications and to consider model interpretation to help human intervention for model prediction confirmation.

1. Introduction

Pathology identification for medical imaging is an important domain of computer vision. It has the capacity to revolutionize medical imaging through more efficient and accurate diagnosis. In particular, it is important to analyse tissues to allow early detection and increase the chance of recovery. In this project, we are going to investigate the possibilities to predict cancers from histopathologic images. Our interest lies in using and comparing classical CNN architectures to solve this binary classification problem with transfer learning.

The input to our algorithm is an histopathological image. We then use a CNN to predict the probability of having a cancer. Since it is a binary classification problem, accuracy and log-loss are obvious metrics to use. However, given the medical context, taking into account the number of number of false positive and false negative is also important. For this reason, we will also consider the *precision* (the probability to have a cancer when diagnosed) and the *recall* (the probability that the cancer will be diagnosed when have cancer). These two metrics are well synthesised by the *F1-score*. Finally, the Receiver Operating Characteristic curves and score (see Figure 3)) are often use in medicine to illustrate the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The principle is to represent, on a graph, and for every possible threshold between 0 and 1, the number of False Positive and False Negative that we would obtain. Hence, we will use these curves and the ROC score (or *ROC curve area*) to assess the quality of our classifiers.

2. Dataset and Features

In this project we are using images from a Kaggle dataset derived from the PatchCamelyon (PCam) dataset, which is derived from the Camelyon16 dataset [1]. The original Camelyon16 dataset contains 400 Hematoxylin and Eosin (H&E) stained histological tissue sections. These images are produced by scanning a complete microscope slide to create a single high-resolution digital file called Whole Slide Image (WSI). These 400 WSI images are used to develop and evaluate algorithms for automated detection of cancer metastasis in digitized lymph node tissue sections. PCam dataset was introduced in [8] to train classifiers on patch-level images extracted from WSI images. WSI images can be of gigapixel resolution whereas PCam patch-level images are 96x96 pixels; hence patch-level classifiers are more computationally tractable. [3] for example argues that detecting cancer is based on cellular-level visual features observed on image patch scale, to justify the use of patch-level classifier. Figure 1 shows some examples of the images in the PCam dataset. PCam contains 327,680 color image patches extracted from a larger dataset with a size of 96x96 pixels with 10x magnification. From these patch-level images derived from larger WSI digital scans, the goal is to identify metastatic cancer tumors. For a 96x96 image, only the center 32x32 of the image is used for determining if there is a tumor or not. The Kaggle dataset, available at `https://www.kaggle.com/c/histopathologic-cancer-detection`, differs from the PCam dataset by having no duplicate images.

In this project, we use and shuffle the train+dev dataset

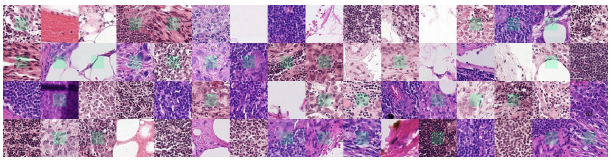

Figure 1. From https://github.com/basveeling/pcam: Example images from PCam. Green boxes indicate tumor tissue in center region, which dictates a positive label.

before subdividing it into a new train/dev/test set. Given the large size of the dataset, we use an approximate 95/2.5/2.5 split (respectively 209,025/5,500/5,500). Our images are pre-processed using the normalization values coming from the pre-training of the CNN model on ImageNet. We also use AdaptiveAvgPool2d to increase the spatial dimension of our image to 224x224 before feeding them in the pre-trained CNN model.

3. Related Work

[3], published in March 2016, proposes to train a Patch-based CNN for WSI images. Following this trend, in June 2018, [8] presents a new large-scale histopathology dataset and proposes a modified DenseNet model that utilizes symmetries inherent to histophatology. DenseNet is presented in [4]. It introduces a dense block which creates a direct connection from a given layer to all subsequent layers by concatenating feature maps. These connections between layers enable better gradient flow and improve feature propagation. [8] proposed to use a modified DenseNet that accounts for rotation and reflection symmetries. A year later [6] proposes an ensemble model on three pre-trained CNNs, namely VGG19, MobileNet, and DenseNet, and obtains state-of-the-art performance, reaching an accuracy of 94.64% on the PCam dataset.

Model	Classification accuracy
P4M-DenseNet	89.9
VGG19	90.84
MobileNetV2	89.09
DenseNet201	87.84

Table 1. From [6]: classification accuracy for some baseline models on PCam dataset

These two approaches gave us a performance benchmark. They both use DenseNet, which can be difficult to train, for instance compared to ResNet. Introduced in 2015, Deep Residual Neural Networks [2] achieved groundbreaking results notably during LSVRC-2015 competition and MSCOCO competition 2015. ResNets are still widely used in research and production with various improved architectures. ResNets were introduced to help train deeper neural networks, which are assumed to have more representation power than shallower models using Kolmogorov-

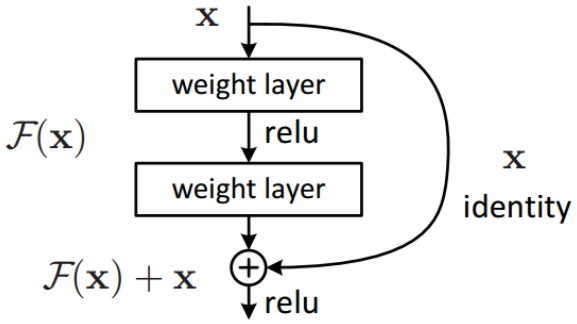

Figure 2. Example of residual block

Arnold representation theorem. But deeper models can be hard to optimize, with gradients vanishing during backprop. To help gradient flow and ensure theoretically that deeper models are at least as good as shallower models, ResNets introduce an identity connection that can skip one or more layers, and introduce residual blocks 2. DenseNet goes beyond this identity connection in ResNet to introduce direct connection from a given layer to all subsequent layers. Theoretically, a dense connection as compared to a residual connection can have several advantages, like better gradient flow to early layers and better parameter efficiency, and could potentially achieve higher accuracy. Still, the original DenseNet implementation is far less memory and speed efficient than ResNet: mainly because the dense connection requires storing much more layer outputs, for the forward and the backward passes. For these reasons, we opted for a ResNet architecture [2].

Our modeling approach is to use a ResNet classifier pre-trained on ImageNet and then fine-tune it on our PCam dataset by progressively unfreezing a few last layers. Because most of the architecture will already be trained, we decided to start with a very deep architecture and opted for the pre-trained implementation of ResNet-101 in PyTorch torchvision library.

4. Methods

4.1. Baseline: ResNet-101 + SVC

ResNet learns linearly separable features before feeding them into a final fully-connected layer. Our baseline model is to use a pre-trained ResNet-101 and re-train the last fully-connected on our specific problem; to verify if these features can be used directly to classify histopathological images. To do so, we had to adapt our input images to be able to feed them to ResNet.

ResNet-101 takes as input 224x224x3 images. After a first convolutional and max pooling layer, ResNet-101 performs four successive sequential blocks, each containing a specific number of Bottleneck blocks applied sequentially.

A Bottleneck block consist into a 1x1 conv2, followed by a 3x3 conv2d that reduces spatial dimension, followed by a 1x1 conv2d that multiples by a factor 4 the image depth, followed by a ReLU activation function. Each conv2d is followed by a batchNorm2d for regularization. Then ResNet-101 performs a global average pool and flattens its ouput to feed it into a 2048 fully-connected layer.

We used AdaptiveAvgPool2d to change the spatial dimensions of our images from 96x96x3 to ResNet input 224x224x3. Then we normalize the input images pixel-wise using mean and standard-deviation values produced during the pre-training of the model. We directly use these values: $mean = [0.485, 0.456, 0.406]$ and $std = [0.229, 0.224, 0.225]$.

Our first was to decide on a good range of values for our model's hyperparameters, especially the learning rate. We performed several sanity checks on a small dataset of 50 images with a batch size of 50, effectively doing a Stochastic Gradient Descent. After checking that our initial loss was effectively going down, we overfitted our small dataset to get almost 100% accuracy after 30 epochs.

The next step would have been to train on a larger dataset. But at that time Google Cloud Platform didn't had any GPUs available to train our model on. Our alternative was then to re define our learning problem to be more computationally efficient. We decided to do a forward pass on a train dataset of 10,000 images and a validation dataset of 200 images, chosen randomly, to produce ResNet's linearly separable features. Then we took these features as input to train a linear classifier, namely a Linear Support Vector Classifier. That way we are still training a "final linear layer" without the burden of doing the forward pass at each epoch. Our learning problem for linear classification is not exactly similar because we are using different loss functions and optimizer procedures. In our modified ResNet architecture we are using BCEWithLogitsLoss, which is a numerically more stable equivalent of applying a Sigmoid and then a Binary Cross Entropy loss. In our Linear Support Vector Classifier, we are using hinge loss. BCEWithLogitsLoss takes a probabilistic approach for defining the loss, whereas hinge loss takes a geometric approach where, for the separable case, the problem is equivalent to finding the closest points in two convex hulls. Still, both BCEWithLogitsLoss and hinge loss define convex optimization problems, hence can be optimized very efficiently, and should result in similar results. Using SVC we are also benefiting from specialized optimizer in scikit-learn that are optimized for this type of learning problem.

Given the small size of our dataset, we find very promising results, summarized in the "Intermediate/Preliminary Results" section, which confirm that our learning problem is legitimate.

4.2. Retraining the Linear Layer on ResNet-18-50-101

On Wednesday May 20th we were able to create a Virtual Machine on GCP. Our first step was to fine-tune our baseline model, a pre-trained ResNet-101 where we re-train the last fully-connected layer, on the whole dataset using a Adam optimizer. As explained above, this problem is equivalent to training a linear classifier on the feature vectors produced by the pre-trained ResNet-101 before the last fully-connected layer. Here, the optimization procedure is different from the SVC approach we used before: our training optimization procedure is now stochastic, and hence it should be less exact than the SVC training optimization procedure. Still, we are now training on a large dataset of 209,025 input features of size 2048; and SVC is thought to not perform well on large dataset because of the associated training complexity. In fact, training a SVC on large dataset can actually be performed using dimensionality reduction methods or approximation methods like stochastic gradient descent, and hence it is less clear that SVC training is more exact than Adam training on the CNN.

From this point forward, we extensively use Tensorboard to monitor our training and dev losses and our ROC curves at each epochs. We set a learning rate of 5e-3 and 30 epochs for training. We evaluate on the dev dataset every 50,000 images.

With a test AUC of 0.96, it clearly confirms the previous promising results. To compare with later models, it is taking around 45 minutes per epoch to train this model.

To explore model reduction, we have implemented the same approach on ResNet-18 and ResNet-50. In the case of ResNet-18, we now have a feature vector of size 512 before our last fully connected layer. We were not able to learn a good model by re-training the last fully-connected layer of ResNet-18: which shows how important are the features learnt in the deeper architecture of ResNet-101, along with a 4x times larger feature vector size. In the case of ResNet-50, we still have a feature vector of size 2048 before our last fully-connected layer. We can successfully train a good model by re-training the last fully-connected layer, with the train and dev losses decaying at the same rate as in ResNet-101 when we where re-training the last fully-connected layer, with slightly more oscillations. We also reach a test AUC of 0.96.

4.3. Unfreezing the last block on ResNet-101

Now, we want to explore more model fine-tuning, and more specifically we want to re-train the fourth sequential layer block of our ResNet-101, which each consists in three bottleneck blocks of three conv2d followed each time by a batchnorm2d, and the last fully connected layer. We kept the same learning parameters. We obtain a very good test AUC of 0.99. It is now taking around 2h20 per epoch to

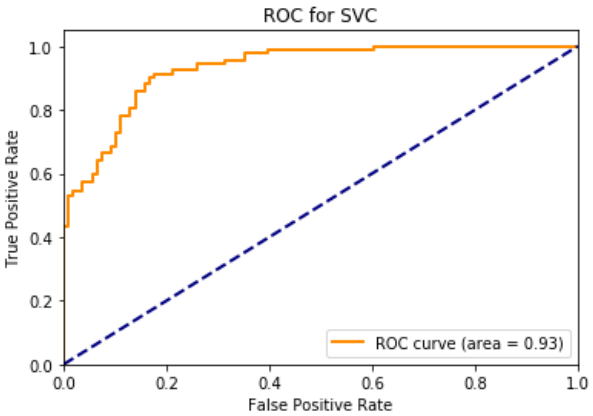

Figure 3. ROC Curve obtained with a Support Vector Classifier, inputs 2048 dimensions from ResNet101, regularization term C=0.1

train this more complex model, and it overfits after a few epochs. Instead of further performance gain, we now want to consider model reduction using ResNet with fewer layers.

5. Experiments/Results/Discussion

5.1. Baseline: ResNet-101 + SVC

The first model we implemented was our baseline. The model consists in using ResNet-101 to pre-compute 2048 features for each image. Then, we use a standard binary classifier. The goal of this model was to start with a simple baseline that would provide a first score for reference. Also, it would give us an idea of the quality of the features from ResNet-101 for our task.

We started this implementation early in the project while we had no GCP VM. Therefore, to compute it locally, we downsampled 10,000 training examples from our training set and compared several binary classifiers. Among all classifiers, Random Forest performed well, but SVC gave better results. Hyperparameters tuning on a val set gave an optimal regularization term $C = 0.1$ which gave the following results (see Table 2). This very simple model gave results close enough to the best score from the original paper, which was a good achievement for a baseline.

Model	NLL	Acc	AUC
P4M-DenseNet	0.260	89.8	96.3
ResNet + SVC	0.334	84	93

Table 2. Comparison of our architecture trained on local machine with state-of-the-art model

5.2. Retraining the Linear Layer on ResNet-18-50-101

Given the good results from the previous model, we considered that ResNet was able to compute relevant features for tumor detection. Moreover, we could observe a large overfit (more than 99% accuracy on the training set) due to the small number of training examples, which also supported this idea. Therefore, our second model was a ResNet network on which we retrained the final output layer. We tried with 3 different ResNet models with respectively 18, 50 and 101 layers and respectively 512, 2048 and 2048 output features.

For the training, we used Adam optimizer with batch size 256 and focused on two main hyperparameters: the learning rate and the L_2 regularization term. To choose the learning rate, we started by overfitting a small training set to find a tradeoff between a large learning rate that would not converge and a small learning rate that would not learn. Eventually, learning rate 2×10^{-3}, 2×10^{-3} and 5×10^{-3} respectively were effective learning rate. Then, we tuned the L_2 regularization term. We tried different values and chose 1×10^{-5} as a good tradeoff.

These models gave the following results (see Table 3)

Model	NLL	AUC
ResNet-18	0.321	93.7
ResNet-50	0.276	95.8
ResNet-101	0.289	95.6

Table 3. Comparison of different ResNet networks where the last FC layer is retrained

ResNet with FC layer training gives results that compete with the original P4M-DenseNet.

5.3. Unfreezing the last block on ResNet-101

Finally, in order to exploit the advantages of transfer learning from ResNet while increasing the complexity of the model, we tried to unfreeze an extra block. For this part, we focused exclusively on ResNet-101 since the model is more complex. We used the same method as in 5.2 to tune hyperparameters and used the learning rate 5×10^{-3} and the L_2 regularization term 1×10^{-5}. Also, we kept used the dropout from ResNet as an extra regularization factor. Finally, in order to improve the score, we performed ensemble learning on this model and obtained the following results (see Table 4):

Model	NLL	Acc.	AUC
P4M-DenseNet	0.260	89.8	96.3
ResNet-101 Ensemble Learning	0.131	94.6	99.0

Table 4. Comparison of our final model with the original paper model

5.4. Results analysis

The previous model gives a very good classifier that optimizes the negative log-likelihood. However, medical applications require to focus on specific metrics other than ac-

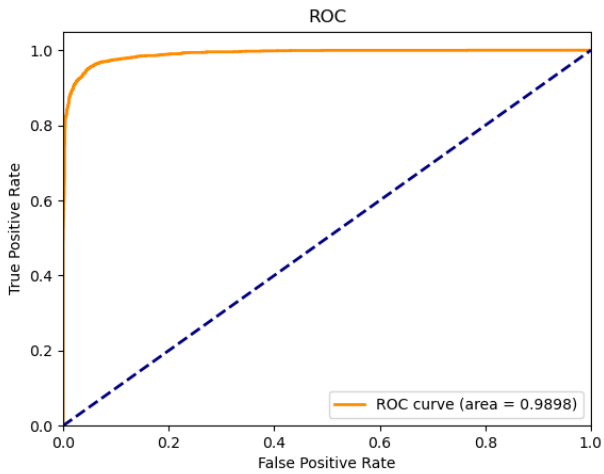

Figure 4. ROC Curve obtained for our final model on Dev Set gives optimal threshold 0.21

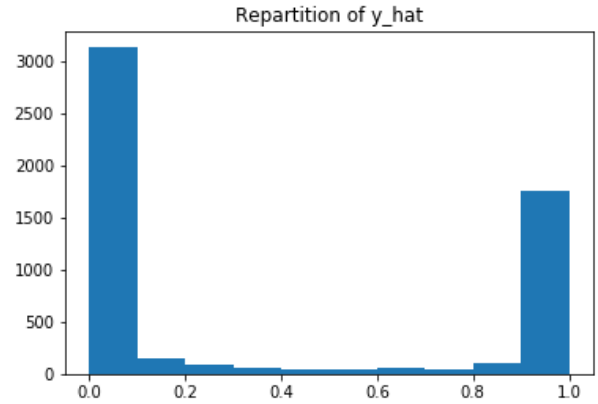

Figure 5. Distribution of output values on the test set

curacy (precision, recall and F1-score mostly, but also the rate of False Negative and False Positive). For this reason, we introduced ROC curves (see Section 1).

Using ROC curves on the dev dataset, we can compute the optimal threshold that would bring us closer to the optimal ROC point $(0, 1)$.

Another way to define the classification threshold would be to use Precision-Recall curves (see Figure 7) and compute the threshold that would optimize the F1-score and bring us closer to the optimal PR point $(1, 1)$.

The first method gives a threshold 0.204 while the second one gives 0.218. Both method give a threshold smaller than 0.5. This is understandable since the dataset is slightly imbalanced (more 0 than 1), therefore, the ROC score requires a smaller threshold than the accuracy. Also, a smaller threshold decreases the False Negative rate, which is important to limit in medicine. Finally, it is interesting to observe the distribution of the predictions on the val set (see Figure 5). The classifier tends to separate much the data in order to maximize the likelihood. Therefore, the classifier is not highly sensitive to the threshold. In the rest of the project, we keep the ROC threshold 0.204 since the original PCam contest used the ROC score as a metric.

With this threshold, we get the following confusion matrix on the test dataset (see Figure 8):

The False Positive Rate was decreased choosing a threshold smaller than 0.5. However, it remains relatively high compared to what could be expected. Considering medical applications, it might be wise to decrease the threshold

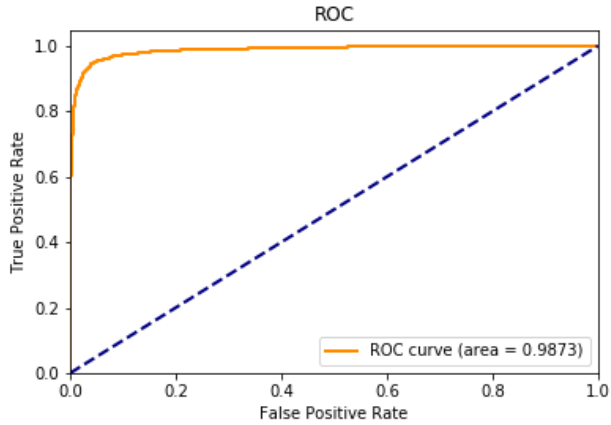

Figure 6. ROC curve on Test Set gives an optimal threshold at 0.21 (v. 0.204 tuned on the Dev Set
).

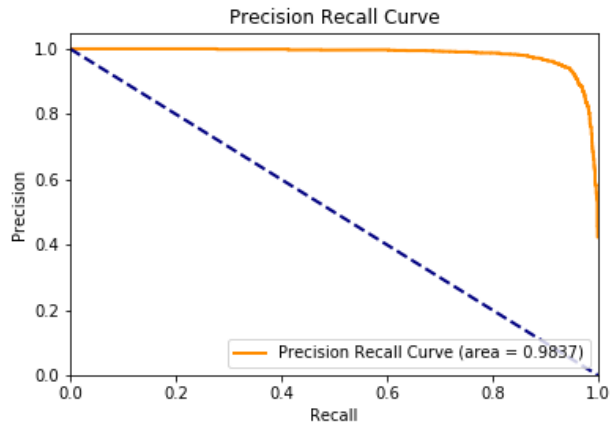

Figure 7. Precision-Recall curve

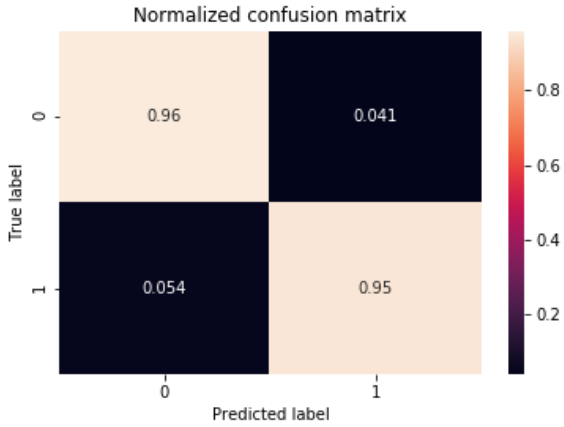

Figure 8. Normalized confusion matrix with tuned threshold 0.204

5.5. Model visualization: saliency maps and grad-CAM

To give a visual interpretation of our models, we have implemented two interpretation methods: saliency maps, introduced in [5], and grad-CAM, introduced in [7]. A saliency map of an image tells us the degree to which each pixel in that image affects the classification score. We implemented the backpropagation saliency map: using backpropagation, we compute the gradient of our loss with respect to each pixel in the image. Class activation map (CAMP) was introduced in [9] to give new interpretations to global average pooling layer as a class-discriminative localization technique. CAM focuses on feature maps before they are averaged using a Global Average Pooling layer and fed in a fully-connected layer. To compute a Class Activation Map, we perform a linear combination of these feature maps, according to an importance score. We obtain an image that can be upsampled to fit the spatial dimension of the input image. Grad-CAM was introduced in [7]. Here the importance score in our linear combination of the feature maps is given by a global average pooling (over the map) of the gradients of the loss with respect to each entry in the feature map. We then apply a ReLU activation function on the linear combination of the feature maps. We can additionally perform guided-backpropagation and perform element-wise multiplication of the guided backprop image and the grad-CAM image to obtain a guided grad-CAM.

Figure 9 shows 5 examples of guided grad-CAM. We only chose images classified as positive. It is in this case that image analysis is relevant. It is notable that the algorithm tends to identify clear areas of the image, which would make confirmation or invalidation easier to a human.

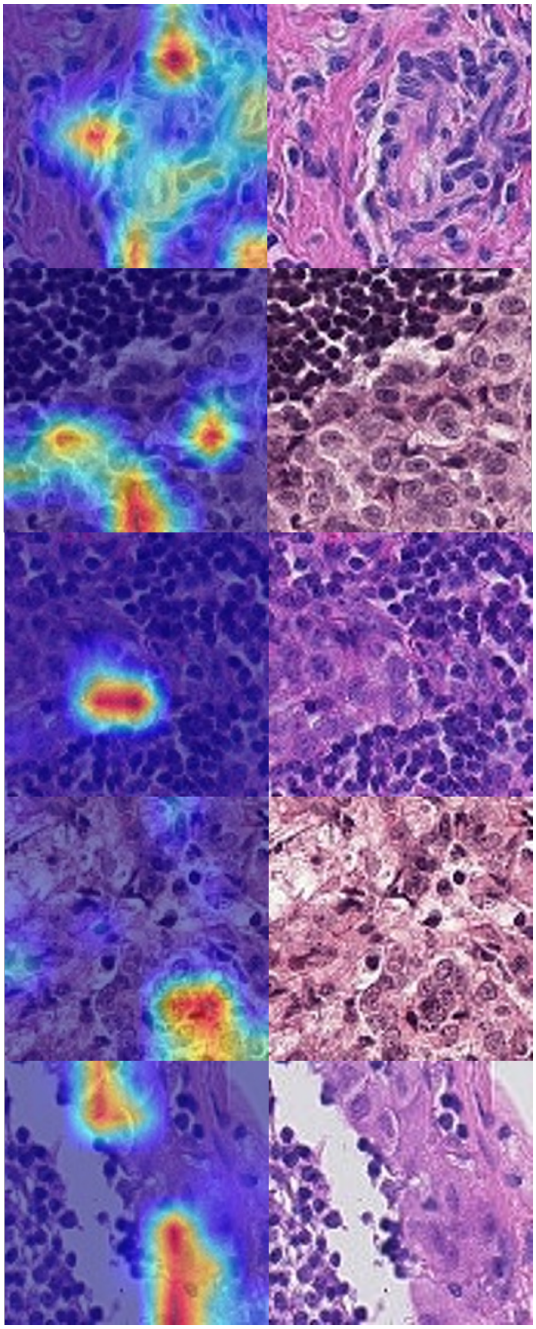

Figure 9. Original image and guided grad-CAM for 5 examples of predicted tumors

6. Conclusion and Future Work

In this project, we were able to use transfer learning to design a very effective algorithm for tumor detection. With ResNet-101 and ensemble learning, we designed an algorithm that gets way better results than the original paper. This project shows that ResNet is able to produce very good 2048-dimensions features for various image classification

tasks. In particular, ResNet-101 is very effective with a lot of data to train this complex model.

Future work on this project would include three main parts. Firstly, it would be important to improve individually the classifiers (by unfreezing an extra block or refining the learning algorithm by more hyperparameters tuning). The second part would be to improve the ensemble learning technique. Finally, the last task would be to use more advanced techniques to compute saliency maps. This is a crucial part of the algorithm if the goal is to assist with diagnosis since it would allow doctors to focus on a tiny part of the image to confirm or deny the classification.

7. Contributions/Acknowledgements

We have worked hand-to-hand together on this project, with equal involvement. Half of the time one of us was implementing our models while the other one was supervising over Zoom the implementation being made and giving advice; and the other half we reverted the roles. There was no external collaborators involved in this project. We used a Virtual Machine on Google Cloud Platform that uses PyTorch 1.4 and 1 x NVIDIA Tesla K80 GPU. The project repository is hosted at: `https://github.com/jvialard/cs-231n-histopathologic-cancer-detection`. All the implementation was made entirely by us from scratch except for grad-CAM where we adapted the implementation from `https://github.com/jacobgil/pytorch-grad-cam`.

References

[1] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes van Diest, Bram van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen A. W. M. van der Laak, , and the CAMELYON16 Consortium. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. *JAMA*, 318(22):2199–2210, 12 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] Le Hou, Dimitris Samaras, Tahsin M. Kurç, Yi Gao, James E. Davis, and Joel H. Saltz. Efficient multiple instance convolutional neural networks for gigapixel resolution image classification. *CoRR*, abs/1504.07947, 2015.

[4] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[5] Andrea Vedaldi Karen Simonyan and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2013.

[6] Michal J. Wesolowski Kevin A. Schneider Ralph Deters Sara Hosseinzadeh Kassani, Peyman Hosseinzadeh Kassani. Classification of histopathological biopsy images using ensemble of deep learning networks. Sept. 2019.

[7] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[8] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. June 2018.

[9] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015.