COMPARING NETWORK ARCHITECTURES FOR LYMPH NODE METASIS CLASSIFICATION

Lisette Boeijenk, Tristan de Boer, Stephan Dooper, Klaus Lux

Radboud University

ABSTRACT

The Patch Camelyon challenge is a lymph node metastasis classification dataset. In this research, we apply different network architectures (capsule networks, convolutional networks, NASNet, and rotation equivariant convolutional networks) on the Patch Camelyon dataset. Here, we achieve results similar to results in literature. Finally, we create an ensemble using eXtreme Gradient Boosting (XGBoost). We demonstrate that the majority of the ensemble is determined by the top-performing NASNet. The ensemble does not generate significantly better predictions than predictions generated by the NASNet.

***Index Terms*—** Patch Camelyon, PCam, RECNN, Capsule Networks, Digital Pathology

1. INTRODUCTION

In recent years pathology, the medical analysis of tissue for diagnostic purposes, has been transformed by technological innovations such as commercially viable whole-slide imaging devices that streamline the process from a tissue sample to a high-quality digital image [1], leading up to a full-blown digital pathology environment. Similarly, the past years have also seen an explosion in research into deep learning, and there has been a lot of scholarly interest in using the technology to automate and improve image analysis in digital pathology [2]. One domain where deep learning would be highly effective is the automated detection of lymph node metastases in women who have been diagnosed with breast cancer: As this procedure is done fairly frequently due to the high incidence rate of breast cancer and very laborious for pathologists, an automated solution could offer large potential benefits. The CAMELYON16 challenge was set up with the intention to leverage this potential: Computer vision researchers from all over the world were provided with stain images from two different labs and tasked with implementing an automated solution capable of beating pathologists at detecting metastases. The results indicated that deep-learning based methods were able to significantly outperform pathologists placed under realistic time pressure, though still behind a pathologist who was given unlimited time [3].

There were relatively high entry barriers in the competition due to the challenge of loading, processing and segmenting whole-slide images. For this reason, Veeling et al. came up with a new challenge based on the original data[4]: In PatchCamelyon, the task setup is easier, only involving the binary classification of small patches of tissue images as to whether there is a metastasis in the center. This setup allows for training a model on a single GPU and does not necessitate the generation of full segmentation maps. It still offers an interesting avenue for inquiry, facilitating the transfer of more general innovations from computer vision to the domain of pathology.

1.1. Contributions

The contributions of this paper are as follows:

- We compare a number of different state-of-the-art network architectures, some of which have never before been applied to lymph node metastasis classification.
- We investigate the use of a data augmentation scheme introduced by [5] with these architectures, finding a generally positive effect of classification performance.
- We use XGBoost to ensemble the best performing models, finding the performance equal to the highest performing model.

2. RELATED WORK

2.1. Related work on Patch Camelyon

Only a few papers have been published on Patch Camelyon (PCam). We summarize results in literature in Table 1.

Veeling et al. report on Rotation Equivariant Convolutional Network (RECNN) or P4M-DenseNet which exploits rotation and reflection symmetries [6]. RECNN extends the group equivariant CNN framework (G-CNN) by adding a $(G \rightarrow \mathbb{Z}^2)$-convolution which transforms feature maps on a group to planar feature maps. CNNs have poor performance when applied to rotation, tilt or any other transformation. Often, this issue is resolved by adding data augmentation, thus adding invariance to the network. Training the network

Network		Metric	
	Name	Test Acc	AUC
[4]	RECNN	0.898	0.963
[4]	DenseNet	0.876	0.955
[7]	S-DenseNet	0.881	-
[8]	Pi	0.8994	-
[8]	Pi+	**0.9036**	-

Table 1. Test results on PCam reported in literature. Results marked with a dash (-) have not been reported in literature.

on augmented data will likely increase the redundancy in weights. RECNN tackles this problem by adding equivariance instead of invariance. An equivariant network will be able to understand rotation and transformations of the data.

Teh and Taylor use self perturbation and contrastive loss on PCam instead of optimizing cross-entropy loss [8]. They propose Pi, which applies self perturbation loss, and Pi+, which applies both perturbation loss and contrastive loss.

Worrall and Welling implement Deep scale-spaces (DSS) on S-DenseNet, which is generalization of CNNs that uses scale-equivariant layers [7]. The network tries to cope with the idea that scale does not always matter when applying classification.

2.2. State-of-the-art Networks

Capsule Networks (CapsNet) are an invention by Hinton [9]. Similarly to RECNN, CapsNet tries to capture one transformation (e.g. rotation or hue) in one capsule. The capsule layer of the CapsNet is a combination of transformations. Using these transformations, a decoder should be able to reconstruct the input image using the features extracted in the capsule layers. The network should perform well on the PCam dataset, as it theoretically would be able to handle H&E staining, rotation, translation, and elastic deformations.

NASNet is currently one of the best performing networks on ImageNet [10]. NASNet applies neural architecture search, and tries to find the optimal architecture using a Recurrent Neural Network (RNN). In this work, we use NASNET-A, the top-performing network architecture generated for ImageNet.

3. DATA AND AUGMENTATION

In this section, we describe the approach taken. First, we describe the dataset, after which we describe the data augmentation that was used.

3.1. Dataset

The PatchCamelyon (PCam) dataset [4] was introduced as a new benchmark for image classification. The dataset consists of 327,680 color images of 96×96 pixels which were extracted from 400 H&E stained WSIs of sentinel lymph node sections from the Camelyon16 dataset [3] at 10× magnification, selected through hard-negative mining. The slides are converted to HSV, blurred and patches are filtered out if the maximum pixel saturation lies below 0.07 to prevent selecting patches of backgrounds. Each image contains a binary label indicating the presence or absence of metastatic tissue in the center 32×32 pixels of the patch. If the center contains at least one pixel of metastatic tissue, the image has a positive label.

The PCam dataset is derived from the Camelyon16 dataset, the samples come from two hospitals in the Netherlands: Radboud University Medical Center (RUMC) and University Medical Center Utrecht (UMCU) and were created using two different scanners [3]. This difference in medical centers and scanners leads to a variation between patches apart from the presence or absence of metastasis, such as a variance in staining color, specimen-level pixel size and sharpness. To make the model robust to these variances we apply different types of augmentation to the training data.

3.2. Augmentation

Image augmentations are an important application to make the set of training images more diverse. Using image augmentations, networks can learn a wider set of variations of images. Tellez et al. [5] researched the effect of stain normalization and stain augmentation, and found that the latter has a significant effect on network performance. One of the problems with having data from different medical centers is that the images can have different stain colors. By performing stain augmentation, we encourage the network architecture to generalize in such a way to incorporate for a different variety of staining colors. We follow the approach in [5] and define several augmentation categories, namely:

- **Basic** includes 90 degrees rotations and mirroring.
- **Morphology** Gaussian blur, Gaussian noise, Elastic deformation, and scaling.
- **Brightness & Contrast** random brightness and contrast perturbations.
- **Hue-Saturation-Value (HSV)** random perturbation randomly shifting the hue and saturation channels in the HSV color space.

Each augmentation category occurs with a probability of $p_{cat} = 0.2$. Within each category, each augmentation occurs with a uniform probability of $p_{aug} = 1/K_{cat}$ where K_{cat} is the number of distinct possible augmentations within the category. See figure 1 for augmentation examples.

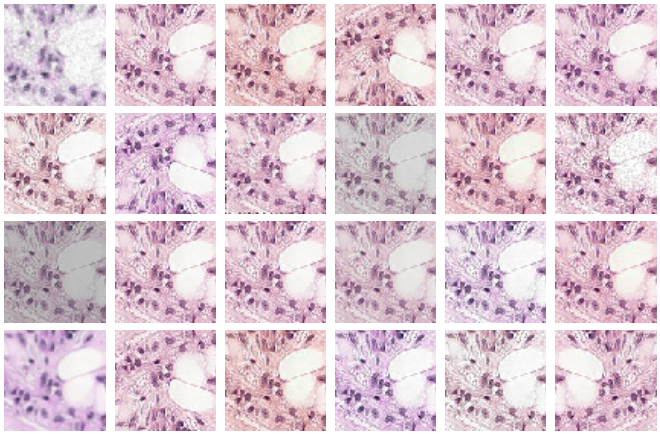


Fig. 1. Several examples of the possible augmentations. The augmentations are all applied on the same image in this example.

4. NETWORKS

In this section, we describe the networks that have been applied to the problem. Each input to each of the networks is a 96×96 pixel image. The output of each of the networks is a fully connected layer with sigmoid activation.

4.1. Convolutional Network

We implement a Convolutional Neural Network (CNN) similar to [5], who performed experiments on similar challenges to PatchCamelyon. The network compromises of four similar blocks and one final block. Each block consists of two convolutional layers, followed by batch normalization (BN) and leaky ReLU activation (LRA). The convolutional layers have 32, 64, 64, 128, 128, 256, 256, and 512 feature maps with 3×3 filters, respectively, and a stride of 2 in the even layers. The final block has a convolutional layer with 512 feature maps and 3×3 filters, followed by global average pooling, 50% dropout, and a dense layer with 512 units, BN and LRA, and a linear dense layer with two outputs, followed by a sigmoid activation. The total number of parameters equals 4,969,985.

4.2. NASNet

NASNet [10] is currently one of the top performing networks on ImageNet. NASNet implements Neural Architecture Search, which samples child networks with different architectures using a recurrent neural network (RNN). In this work we use the network defined as NASNet-A (6 @ 4032) in [10], however we changed the input size from 331×331 to 96×96 to match the input size of the current challenge. We also removed the top layers from the network and replaced it by a block with global average pooling, a dense layer with 512 units, and a prediction layer with 1 unit and sigmoid activation. This network has 86,982,227 parameters.

4.3. CapsNet

Capsule networks (CapsNet) were proposed in [9]. A major disadvantage of CNN features is that positional information is lost. The authors argued that it was important to keep such information present. For this reason, they proposed to use a capsule network, which is equivariant. This enables the network to detect simple transformations of objects, such as rotations or flips. For the full explanation of this network and additional information we refer the reader to [9, 11].

The CapsNet architecture used resembles the CapsNet architecture proposed by [9] and consists of a convolution block and a CapsNet block. The convolution block consists of six 2-dimensional convolutional layers of sizes 32, 64, 64, 128, 128, and 256 filters. All convolutional layers have 3×3 kernels with valid padding and use ReLU activation. All of the even layers have stride 2 and the odd layers have stride 1. This will slightly increase the receptive field of the network. As an additional experiment, we train an additional CapsNet that incorporates Batch Normalization after every convolution layer.

The CapNet block consists of a primary capsule layer (PrimaryCaps) and digit capsule layer (DigitCaps). The PrimaryCaps is a convolution capsule with 32 channels of 8 dimensional convolution capsules. Each primary capsule consists of 8 convolutional units with a 9×9 kernel, which applies valid padding, and a stride of 2. PrimaryCaps inputs the final convolutional layer of the convolutional block with dimensions $256 \times (3 \times 3)$. The output of PrimaryCaps with dimension $32 \times 8 \times (6 \times 6)$ serves as input for DigitCaps, which has one 16 dimensional capsule layer for every class. Only one class is predicted, thus DigitCaps consists of only one 16 dimensional capsule layer.

The final CapsNet has $12,848,000$ parameters. The network with Batch Normalization has $12,998,272$ parameters.

We furthermore implement a decoder. The decoder is used as regularizer, and may force the network to learn useful features that can be used to reconstruct the original input. It takes the 16 dimensional capsule layer as input and tries to reconstruct the original input based on learned features. The decoder consists of fully connected layers, with respectively 256 and $27,648$ neurons. We reshape the final layer of $27,648$ neurons to $96 \times 96 \times 3$, which is the size of the input shape, on which we apply the sigmoid activation function.

4.4. Rotation Equivariant CNNs

Introduced by [4], Rotation Equivarant CNNs build upon the DenseNet architecture introduced by [12] by adding a rotation-equivariant convolution operation. By convolving intermediate representations not only with a number of feature maps, but also with their rotated and optionally mirrored

variants, feeding rotated versions of the same image will cause the outputs of the network for these images to be rotated variants of each other. This property is known as rotation equivariance and hypothesized to be useful in the domain of pathology, as structures in pathology such as tumors are thought to be equivariant to rotation as well. We experimented with this architecture as it was designed specifically for the PatchCamelyon challenge and was reported to perform quite well at an AUC of 0.963. The architecture has a relatively high number of hyperparameters, few of which have any default values. For some of them the authors specify values used for the experiments, specifically the number of blocks is set to 5, the number of layers per block is set to 1 and the D4 convolution group performs best, used with a growth rate of 8. For other parameters, no choices are mentioned, so we tuned them based on how many weights the resulting architecture had, as that number was given by the authors in paper. Based on these experiments, we chose to include bottleneck blocks and a fully connected layer at the top. The reduction factor within the transition block was determined to be 0.33. The resulting network has 115,687 parameters, which is reasonably close to the 119,000 parameters mentioned by the authors.

4.5. Network Ensemble

Predictions can be often improved by combining predictions from predictors trained on differently initialized predictors. There exist many methods on how to combine multiple networks, simple examples include averaging, majority voting, or taking the maximum of the predictions. Here, we use the predictions from the validation set to generate a classifier for determining the ensemble output. This is done using eXtreme Gradient Boosting (XGBoost) [13]. XGBoost ensembles predictions generated by different decision trees, which are trained using gradient boosting fashion. This method allows to learn complex correlations across predictions and between models.

The ensemble was trained using the validation data from the CapsNet, CapsNet with batch normalization, ConvNet, NasNet and RECNN networks. Each of the networks was trained with augmented data.

To tune the parameters, we perform cross validation on the validation dataset. The cross validation is used to determine optimal hyperparameters for the XGBoost classifier using grid search. We list the values used for grid search in table 2 where the optimal hyperparameters for the ensemble are highlighted. We set the number of decision trees to 1000.

5. EXPERIMENTS

The loss of the network is computed using binary crossentropy, as seen in Equation 1. Next, we use Adam [14] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and set the initial learning rate to 1×10^{-3}. We reduce the learning rate with factor 0.1 when the validation loss stagnates, using a patience value of 4 epochs, which means that the learning rate will again be reduced after 4 epochs after the last reduction. The lower bound of the learning rate is set to $1e-5$. All of the networks are trained with batch size $B = 32$ and training is terminated if there is no overall improvement based on the validation loss within 15 epochs.

Parameter	Values
Max Depth	$[3, \mathbf{4}, 5, 6, 8, 10]$
Child Weight	$[\mathbf{1}, 2, 3]$
Learning Rate	$[0.1, 0.15, 0.2, \mathbf{0.25}, 0.3]$
γ	$[0, 0.1, \mathbf{0.2}]$

Table 2. Grid Search values for determining optimal parameters for XGBoost. The learned parameters for the ensemble are highlighted.

Network	Validation accuracy	
	No Augmentation	Augmentation
ConvNet	0.8345	0.8682
RECNN	0.8269	0.8915

Table 3. Effect of data augmentation on classification performance of two architectures.

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=0}^{N} y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i) \quad (1)$$

We apply sigmoid activation on the output of every network.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (2)$$

Finally, the networks all use the augmentation defined in section 3.2. In order to validate the use of augmentation for our problem, we the validation scores of some of the networks with and without augmentation, which is shown in table 3.

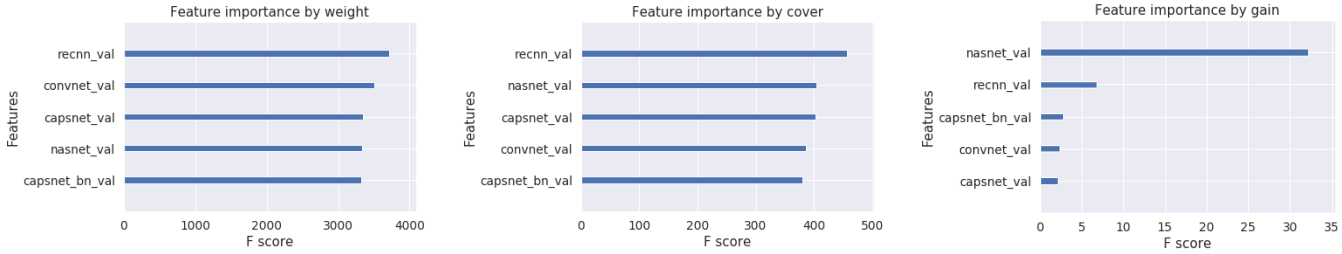
6. RESULTS

The accuracy and AUC results on the test set are reported in Table 4. We observe that among the individual networks that we experimented with, the NASNet has the highest AUC (0.9623), followed by the RECNN (0.9574), ConvNet (0.8682), and CapsNets (0.9323) and (0.9144), respectively.

The fact that the NASNet performs well even if it has an excessive amount of total parameters can be explained due to the extensive regularization techniques that were used in this architecture. Most notable the authors of the NASNet [10] report that their scheduledDropPath regularization technique

Network	Configuration		Metric	
	Epochs	Params	Test Acc	AUC
CapsNet	15	12,848,000	0.8390	0.9323
CapsNet BN	15	12,998,272	0.8405	0.9144
ConvNet*	15	4,969,985	0.8682	0.9489
NASNet	15	86,982,227	0.8924	**0.9623**
RECNN	15	115,687	0.8926	0.9574
XGBoost Ensemble			**0.9009**	**0.9623**
P4M-DenseNet (RECNN) [6]			.8980	**0.963**
Pi+ [7]			**0.9036**	-

Table 4. Test results from the different models based on accuracy and AUC. All of the models are trained in the way specified in section 5



(a) (b) (c)

Fig. 2. Importance of each network in the XGBoost ensemble calculated by (a) the number of times a feature appears in a tree, (b) the average number of samples affected by splits which use the feature and (c) the average gain of splits which use the feature.

allows the NASNet to generalize well. The CapsNets have the worst overall performance among the models. Initially, we expected this network to perform better than at least the ConvNet, as they take into account orientation, which should allow them to infer affine transformations such as rotations and flips. A possible reason for the lower performance is that the capsule network architectures are still quite new, and further research is needed to optimize them, see for instance [15] for a possible improvement. One surprising result of the RECNN is that, although it has far fewer parameters than all of the other models in several orders of magnitude, it performs second best in terms of AUC, and best in terms of accuracy. The final ensemble which mixes the predictions from all the models achieves an overall test AUC of 0.9623, which does not improve on the NASNet.

Figure 2 displays the importance of the used features in the XGBoost models. Figure 2a shows that every network is used almost equally in the decision trees. Figure 2b shows the number of times a network is used as split node. Here, we can conclude that the RECNN decision is used more often than CapsNet BN. 2c shows us the information gained by using NASNet. This means that NASNet is often an important factor in determining the ensemble prediction.

7. CONCLUSION

In this paper, we investigated the use of deep learning for the classification of Patch Camelyon, a challenge in histopathologic scans of lymph node sections. Several neural network architectures were proposed and tested on test accuracy and AUC performance. We were able to achieve decent results on state-of-the-art architectures, including RECNN and CapsNet, with results comparable to results reported in literature. Furthermore, we were able to achieve results similar to state-of-the-art using NASNet. Finally, we created an ensemble using eXtreme Gradient Boosting (XGBoost) of all trained networks, which achieved a final score of 0.9623 on the test set. We demonstrated that the XGBoost classifier used for ensembling primarily relied on NASNet predictions.

For future work, several other approaches could be tried. Firstly, the CapsNet architecture is still in its infancy and several improvements could be made to it. For example, changing the receptive field of the primary capsule layer could have impact on the performance. Secondly, we have not yet tried to experiment with the proposed architectures on the much more difficult task of segmentation, which is the original objective in the Camelyon challenge. This task is much harder due to the large input image, and the fact that an output map has to be generated, rather than a single value.

8. REFERENCES

[1] Liron Pantanowitz, Ashish Sharma, Alexis B. Carter, Tahsin Kurc, Alan Sussman, and Joel Saltz, "Twenty Years of Digital Pathology: An Overview of the Road Travelled, What is on the Horizon, and the Emergence of Vendor-Neutral Archives," *Journal of Pathology Informatics*, vol. 9, Nov. 2018.

[2] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Snchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, Dec. 2017.

[3] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes Van Diest, Bram Van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen AWM Van Der Laak, Meyke Hermsen, Quirine F Manson, Maschenka Balkenhol, et al., "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer," *Jama*, vol. 318, no. 22, pp. 2199–2210, 2017.

[4] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling, "Rotation equivariant cnns for digital pathology," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2018, pp. 210–218.

[5] David Tellez, Geert Litjens, Peter Bandi, Wouter Bulten, John-Melle Bokhorst, Francesco Ciompi, and Jeroen van der Laak, "Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology," *arXiv preprint arXiv:1902.06543*, 2019.

[6] Jasper Linmans, Jim Winkens, Bastiaan S. Veeling, Taco S. Cohen, and Max Welling, "Sample efficient semantic segmentation using rotation equivariant convolutional networks," *CoRR*, vol. abs/1807.00583, 2018.

[7] Daniel E Worrall and Max Welling, "Deep scale-spaces: Equivariance over scale," *arXiv preprint arXiv:1905.11697*, 2019.

[8] Eu Wern Teh and Graham W. Taylor, "Metric learning for patch classification in digital pathology," in *International Conference on Medical Imaging with Deep Learning -- Extended Abstract Track*, London, United Kingdom, 08–10 Jul 2019.

[9] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., pp. 3856–3866. Curran Associates, Inc., 2017.

[10] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le, "Learning transferable architectures for scalable image recognition," *CoRR*, vol. abs/1707.07012, 2017.

[11] Aryan Mobiny and Hien Van Nguyen, "Fast capsnet for lung cancer screening," in *MICCAI*, 2018.

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[13] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[14] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[15] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst, "Matrix capsules with EM routing," in *International Conference on Learning Representations*, 2018.

A. SOURCE CODE

Source code and installation instructions can be found on the Github page: `https://github.com/stephandooper/ismi/tree/master/Project/`