Histopathologic Cancer Detection

Nicolas Alder
Master Data Engineering
Hasso Plattner Institute
University of Potsdam

Eric Fischer
Master Data Engineering
Hasso Plattner Institute
University of Potsdam

Erik Langenhan
Master IT Systems Engineering
Hasso Plattner Institute
University of Potsdam

Nataniel Müller
Master Digital Health
Hasso Plattner Institute
University of Potsdam

Christian Warmuth
Master Data Engineering
Hasso Plattner Institute
University of Potsdam

Simon Witzke
Master Data Engineering
Hasso Plattner Institute
University of Potsdam

Abstract

Sentinel lymph node biopsy and the analysis of histopathological whole slide images are essential for early-detection, staging, and therapy of cancer. Nevertheless, the classification of non-malignant and malignant tissue can be time-consuming, repetitive, and often challenging for pathologists. Deep Learning approaches have recently gained popularity in digital histopathology due to high prediction accuracies with improved objectivity and efficiency. However, in the context of clinical practice, it is valuable to assess and compare the performance of various state-of-the-art Deep Learning architectures. This paper provides an in-depth overview of the performance of VGGNets, ResNets, and DenseNets on a reduced Patch-Camelyon dataset, which contains 220,025 whole slide images of tissue scans. Although stain normalization did not improve performance, the best networks achieved over 99% AUROC. A pretrained ResNet34 achieved the best AUROC of 99.53%, showing that Convolutional Neural Networks are a suitable method to detect cancer in lymph node tissue and could support pathologists' workflow significantly.

1 Introduction

Mortality rates associated with cancer in Germany have been steadily declining since the 1990s. However, the absolute number of new cases has almost doubled since the 1970s due to an increased overall life expectancy and an aging population. In 2016, the incidence of all cancer types in Germany was 229,900 with a 5-year prevalence of 791,770 [1].

Early-stage cancer frequently metastasizes from a primary tumor to regional lymph nodes, associated with a reduced survival rate [2]. The sentinel lymph node biopsy technique for assessing cancer spread involves a time-consuming, multi-section lymph node inspection. 60-70% of the sentinel lymph nodes do not contain any metastasis, thus making it tedious in clinical practice [3]. Furthermore, the importance of early metastasis detection in lymph nodes is indicated by a reduced 5-year survival rate of stage II (82,5%) without lymph node metastases compared to stage III (59.5%) with lymph node metastasis [4].

The gold standard of diagnosing many cancer types is the microscopic examination of hematoxylin and eosin (H&E)-stained biopsy specimen [5]. Histology slide analysis extracts common image features generally divided into four categories: morphometry (area, size, boundary & shape), topology (structural, i.e., Voronoi diagram), intensity/color, and textures [6]. Furthermore, increasing incidence and patient-specific treatment options, combined with a large number of slides assessed

by pathologists in clinical routine, has made the diagnosis and staging of cancer progressively more complex [3]. Advances in the digital processing of histopathologic using whole slide imaging (WSI) allow for large scale analysis and processing and paved the way for the application of Deep Learning models requiring large datasets [7].

The detection of histopathologic cancer is a specific, complex, and repetitive task. Especially Deep Convolutional Neural Networks (CNNs) have gained popularity achieving accuracies comparable to medical professionals [8, 3, 9, 10]. Moreover, recent studies indicate that pathologists supported by Deep-Learning-based assistants exhibit an increase in their detection performance (i.e., accuracy, sensitivity, and time effort for one image) [11, 12]. To achieve high accuracies, the architecture of the CNN is critical. However, researchers often only examine one specific architecture that worked best for their dataset. This paper examines different CNN-architectures against the Kaggle PCam dataset[1] and evaluates their potential strengths and limitations. Moreover, the effect of pretraining and domain-specific normalization (i.e., staining normalization) is assessed. Despite recent achievements in this area, major challenges remain, such as noisy ground truth labels, stain variance across datasets and samples, unbalanced classes, and often limited availability of labeled data [13].

We define the following problem statement: The detection of malignant tissue in WSI is time-consuming and tiresome in clinical histopathology. The application of Deep Learning methods to automatically classify malignant and non-malignant lymph node tissue would significantly decrease the time spent on diagnosis. In this paper, we evaluate the suitability of state-of-the-art conventional Deep Learning architectures.

The paper is structured as follows: In Section 2, we review related work. Section 3 explains the structure of the evaluated architectures, our implementation, data preprocessing, and experiment setup. In Section 4, we summarize the results of the architectures. Section 5 discusses the results and outlines future work. Last, we conclude in Section 6.

2 Related Work

Computer-Assisted Diagnosis (CAD) methods for histopathologic cancer detection can be subdivided into (1) traditional machine learning approaches, based on structure detection and texture extraction, and (2) more recent Deep Learning approaches [14]. Traditional classification approaches such as Random Forest with 18 features obtained 83% accuracy in a 7-class classification of prostate tissue [15]. Support Vector Machines (SVM) were used to classify hyperspectral colon tissue cells yielding 87.1% accuracy [16]. Recent works tackling automatic histopathologic cancer detection have been proposed.

The most successful approaches are based on Deep Neural Networks which, in contrast to traditional methods, outperform pathologists [17]. In 2016, Bayramoglue et al. and Spanhol et al. used basic CNNs (i.e., three convolutions followed by pooling and a non-linearity) to detect breast cancer in histopathologic images on the BreaKHis dataset, achieving accuracies of up to 90% [8, 18]. Chakraborty et al. [19] implemented a Dual-Channel Residual Convolutional Neural Network on the same dataset with stain decomposition, achieving an overall accuracy of 96.48%, average recall of 95.72%, and an average precision of 95.92% respectively. Jaiswal et al. [20] applied similar architectures including VGGNet16 (97.45%), SE-ResNet101 (97.83%), and DenseNet201 (97.94%) on the PCam dataset. Recently, Zhai et al. [21] reviewed several modern CNN-architectures such as InceptionV3, DenseNet201, and ResNet50 on the PCam dataset achieving accuracies of up to 93.59%. However, no other metric besides accuracy was reported. We infer that there is a need for a thorough evaluation of modern CNN-architectures and, applied to histopathologic cancer detection on the PCam dataset, evaluation of more metrics such as recall, precision, F_1 or Area Under the Receiver Operating Characteristics (AUROC).

3 Approach

After providing an overview on the given dataset and preprocessing steps, we introduce the implemented architectures and describe the general experiment setup.

[1] `https://www.kaggle.com/c/histopathologic-cancer-detection`

3.1 Data Foundation and Preprocessing

We utilized a slightly modified version of the PatchCamelyon (PCam) dataset which was in use for a Kaggle competition [22]. This dataset consists of 220,025 images of tissue from lymph node sections, 96x96 pixels, and 27,935 bytes each. Figure 1 shows that 130,908 of these pictures show no signs of cancer, represented by class 0. The other 89,117 pictures that show signs of cancer are represented by class 1.

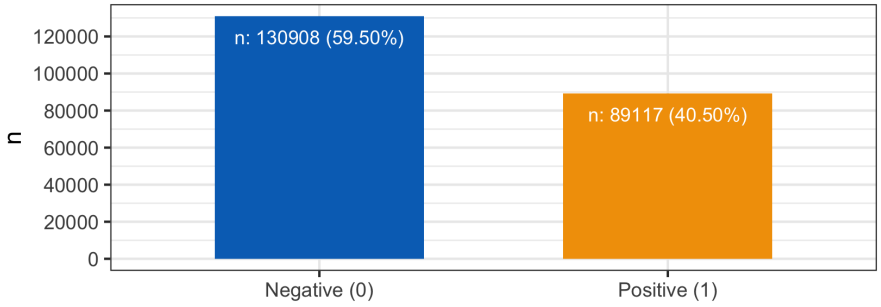
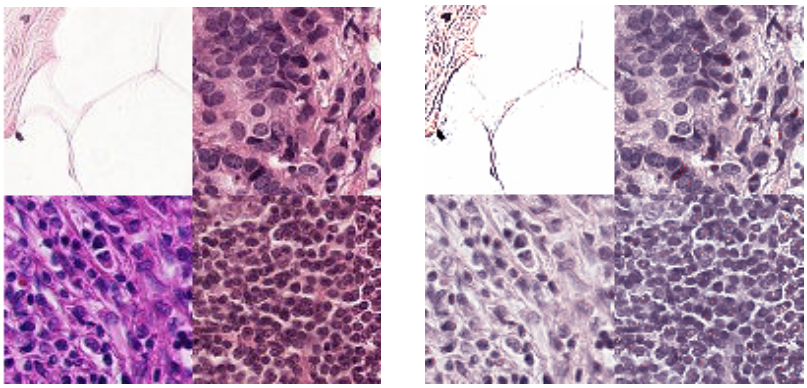


Figure 1: Label Distribution in Training Data

In the process of deriving the sample tissues, the hematoxylin and eosin (H&E)-staining was used for the dataset at hand to analyze different biological substances with different selective affinities [22, 23]. The stained slides are analyzed using a microscope while being illuminated from below. The stain vector is the proportion of each wavelength absorbed by the stained slide. This stain vector can vary between the different stains used in the process, but it can also vary considerably for the same stain depending on factors such as the manufacturer, storage conditions before use, and the application method itself [24, 25].

For computer-driven analysis, a normalization procedure is applied as feature measurements might origin from different laboratories and different preparation introducing systematic biases [23]. In order to account for the inconsistencies in the preparation of histology slides, we applied staining normalization.



(a) Original (b) Normalized

Figure 2: Comparison of Non-Normalized and Normalized Images

We pursued the approach of Macenko et. al. [23] and their SVD-gedesic method for obtaining normalized stain vectors. We used an existing Python implementation[2]. Figure 2 shows four exemplary tissue slides in original and normalized.

The normalization was done before loading the images. This procedure was chosen to reduce the image-loading time as normalizing all images in the dataset required multiple hours. Due to the

[2] www.github.com/schaugf/HEnorm_python

nature of the normalization algorithms, the eigenvalues of the stain-value-matrices were computed. For some matrices, the eigenvalue-decomposition did not converge. As this only accounted for 418 out of 220,025 pictures, we did not include these pictures in the normalized dataset.

3.2 Architectures

We selected LeNet, VGGNet, ResNet, and DenseNet as the architectures to tackle the task. The selection of the LeNet is motivated to additionally incorporate a very basic architecture and simple model. The latter three are consistent with successful architectures adopted from related work. The models were published in the same order as written here in the context of the benchmarking dataset ImageNet [26]. They reflect the domain-independent historical development of increasingly performant Deep Learning architectures for image recognition. These models allow us to examine if newer architectures that score better on ImageNet [26] also score better in the recognition of histopathologic cancer compared to previous architectures since they address deficits of their precedents. We briefly describe the implemented architectures and their characteristics in the following.

LeNet5 [27] uses two convolutional layers, each followed by an average-pooling operation. Finally, three fully connected dense layers with a sigmoid activation function reduce the feature maps to the desired output dimension. We use max-pooling and ReLUs instead of the average-pooling and sigmoid activation function, as these give better results [28]. As it is the most basic network, we expect the worst results here.

Simonyan and Zisserman [29] evaluated different architectures, referred to as Visual Geometry Group Networks (VGGNet) in 2016. They are characterized by very small convolution filters (3x3), but many (deep) layers. Their main contribution is the observation that deeper convolution filters achieve better results than a few large convolution filters. We use VGGNet11 with low depth and VGGNet19 with greater depth as they suggest performance gains.

Despite deeper architectures promise better results, they come with new challenges. He et al. [30] pose the following question: "Is learning better networks as easy as stacking more layers?" Simonyan and Zisserman [29] observe improved results with deeper networks, but at the same time, with increasing depth, degradation [30], vanishing gradients, a diminishing forward flow of information (or diminishing feature reuse), and a great increase of training time occur [31]. VGGNet11 and VGGNet19 are the only architectures that are likely to show this behavior, as the LeNet5 is too shallow and ResNets and DenseNets counteract these phenomena in their design. However, as He et al. [30] observed those appearances for a 20-layer model after more than 30,000 epochs, we do not expect this a problem for VGGNets.

Residual Networks (ResNet) [30] are chosen to incorporate a state-of-the-art architecture. It addresses, in particular, the problem of degradation. The key elements are the so-called residual blocks. In contrast to an architecture in which layers are stacked on top of each other in a plain feedforward fashion, ResNets contain shortcut connections. These additional connections not only transport information into the subsequent layer but also skip layers, thus transporting the same information (identity mapping) to later layers (forward propagation). This allows for easier identification of useful gradient directions for later layers (backpropagation) thus optimizing the model with respect to feature reuse as well as vanishing gradients [31]. Compared to identical networks without shortcut connections, it can be empirically observed that test error and degradation diminish [30]. This utilizes deeper networks for improved accuracy. We use a shallower ResNet121 and a deeper ResNet201. ResNets are expected to be at least as good as VGGNets but may even profit further from their improved gradient flow.

The most recent employed architecture is the Densely Connected Convolutional Network (DenseNet) presented by Huang et al. in 2017 [32]. The authors further utilize the underlying principle of ResNets. DenseNets connect each layer with each subsequent layer. Each layer receives not only the input from its previous layer but the input from all previous layers. Unlike ResNets, these previous features are not passed to a layer by an add operation, but by concatenation. The final classifier in the last layer of the DenseNet decides on the basis of all feature maps that exist in the network and not only on the last one(s) as implemented in ResNets. We evaluate a shallower DenseNet121 and a deeper DenseNet201 for our task. We expect the DenseNet architecture to perform best of all chosen architectures as it incorporates all formerly mentioned design principles.

3.3 Experiment Setup

We split the data into a training and test set with 176,020 (80%) and 44,005 (20%) images respectively. We considered this number of images to be sufficient to train and evaluate our models with a pure train-test split. This way, we avoided using Cross-Validation and thus the related increase in training time that comes with k-fold iterations. To load the data, we implemented a custom dataset class based on the PyTorch Dataset. Therefore, we are compatible with torchvision image transformations such as random flipping and rotation. We apply these transformations on our training data to simulate a greater variety of images during model training. Further, we can normalize the RGB channels using a z-transformation which is necessary for pretrained models. Moreover, we implemented an in-memory option to increase image loading performance.

We specify our neural networks as new classes based on pre-implemented PyTorch architectures. We adapted these architectures to match our binary classification task by adjusting their classification layer to a single output. For model training, we use skorch [33], a scikit-learn [34] compatible wrapper for PyTorch. Skorch provides a high-level wrapper for train and test loops that we supply with our model and hyperparameters such as an optimizer, epoch numbers, learning rate, and batch size. We also pass the wrapper callbacks specifying our evaluation metrics: accuracy, measuring the proportion of correctly identified images over all images; precision, measuring the proportion of correctly detected cancer images over all images that have been predicted as cancer; recall, measuring the fraction of correctly detected cancer images over all cancer images; F_1, the harmonic mean of precision and recall, thus considering both false positive and false negative cancer predictions, and AUROC. The ROC curve sets the recall in proportion to the false positive rate for different classification thresholds of the models. The AUROC summarizes the area under this curve in one number and allows a comparison of this curve for different models. We chose to collect these metrics as they are standard for binary classification tasks and make our results comparable to other work. To track all trained models and their hyperparameters and metrics, we connected our training setup to neptune.ai[3], an experiment management tool. Thus, any trained model can be easily accessed and used for prediction.

In order to train models with both Google Colab[4] (Nvidia Tesla K80 GPU) and on an enterprise-grade server (four Nvidia Tesla V100 GPUs), we implemented two different training scripts that either accept parameters as command line arguments or pass them using callable functions.

In total, we conducted 150 experiments with 30 epochs over all architectures. We used binary cross-entropy logit loss and an Adam optimizer which tends to lead to faster convergence [35] for all experiments. The LeNet, DenseNets, and ResNets were trained with a batch size of 128, learning rate of 0.01, Adam optimizer, exponential learning rate scheduler with γ of 0.9. For VGGNets, we had to reduce the learning rate to 0.001 and use an Adamax optimizer to obtain usable results. We refrained from extensive hyperparameter tuning (e.g., using GridSearch) as this significantly increased training times. Instead, we selected our hyperparameters based on related work.

Initially, we trained LeNet, VGGNet, ResNet, and DenseNet on a reduced training dataset (20,000 images) without stain normalization. This gives an insight into the performance of the different architectures on a relatively small dataset. We then evaluated these architectures on the full training dataset with- and without stain normalization (cf. subsection 3.1).

Further, we evaluated the performance of ResNet and DenseNet models that were pretrained on the Imagenet dataset for the complete training dataset. For all pretrained experiments, we only retrained the last layers and thus used a stepwise learning rate scheduler instead of the exponential scheduler. For ResNets, we also adjusted the learning rate to 0.001. Due to limited time, we did not evaluate the pretrained models on the reduced or stain normalized datasets. For pretrained DenseNet we used the DenseNet121 and DenseNet201. For pretrained ResNet we used ResNet34 and ResNet101.

4 Results

When reporting model performance, without further specification, we refer to the weighted mean of the respective metrics over all runs. The exact results of the architectures and models can be

[3] `https://neptune.ai`

[4] `https://colab.research.google.com`

viewed in the Appendix in Table 2, 3, 4, 5 and 7. LeNet5 merely achieves an accuracy of 59.09% (the majority class distribution) and is hence not suited for this task. Therefore, we exclude it from further analysis.

For the non-normalized full dataset, we observe that a non-pretrained DenseNet201 with a learning rate of 0.01 yields the highest accuracy (97.65%), F_1 (97.09%) and recall (97.57%). The best AUROC value (99.53%) is scored for a pretrained ResNet34 model. All metrics and hyperparameters of the best runs are recorded in Table 1. The worst performances were observed for a non-pretrained ResNet152 (Table 6).

	DenseNet201	**DenseNet121**	**DenseNet201**	**ResNet34**
Epochs	30	30	30	30
Learning Rate	0.01	0.01	0.01	0.001
Avg Duration/Epoch in s	652	250	435	697
Full Data	full	full	full	full
Normalized	no	no.	no	no
Optimizer	Adam	Adam	Adam	Adam
Pretrained	no	no	no	yes
Test Accuracy	97.65	97.47	97.19	97.11
Test F_1	97.09	96.85	96.57	96.43
Test Precision	97.28	97.47	95.58	96.45
Test Recall	96.91	96.24	97.57	96.41
Test AUROC	97.54	97.27	97.25	99.53

Table 1: Best test metrics (in %)

To compare entire architectures independently of specific model configurations (e.g., VGGNet with ResNet), we average the evaluation results over multiple experiment runs with identical hyperparameters configurations. We observe similar performances when comparing the weighted means for the non-pretrained network architectures VGGNets, ResNets, and DenseNets on the full dataset. Figure 3 shows that the AUROC metric is very close for the architectures at the end of 30 epoch runs. However, if using normalized data for the different models, the overall performance slightly decreases. If using pretrained networks on the non-normalized, full dataset, a slight performance increase (≈1%) for ResNets can be observed, while DenseNets experience a slight performance decrease (≈1-2%). If using only a partial subset of training data with 20,000 samples, the performance of all architectures decreases significantly. However, AUROC only decreases by less than 3-5%, while other metrics decrease more than 10% (see Figure 7 in Appendix). In general, we observe that DenseNets have the best performance on a partial subset of the non-normalized data.

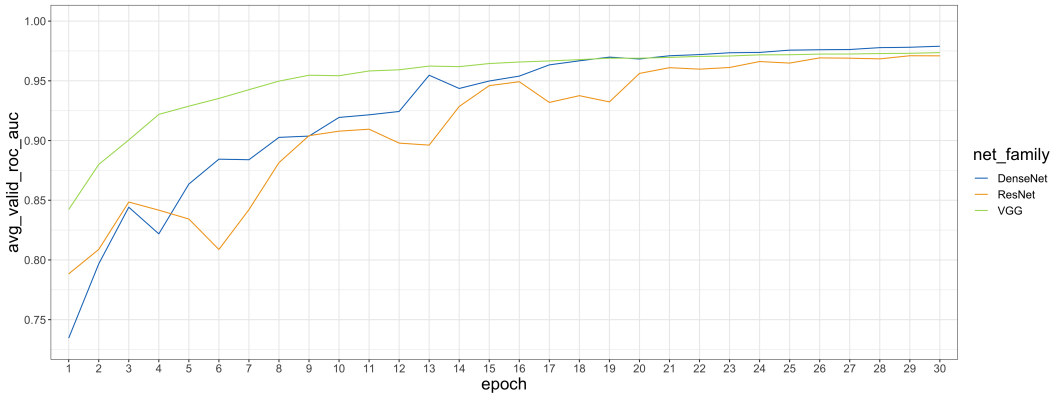


Figure 3: Weighted mean AUROC plot of network architectures

The difference between using a shallow versus a deep variant of the architecture appears neglectable. The VGGNet19 only shows very little performance gains compared to a VGGNet11. Furthermore, we do not observe vanishing gradients, diminished information flows, or degradation in the VGGNet as formerly expected. The ResNet18 and ResNet152 as well as the DenseNet121 and DenseNet201

do not show noteworthy differences. Only the pretrained DenseNet201 performs slightly better than the pretrained DenseNet121 on all metrics despite having an equally good recall for both models.

5 Discussion of Results

VGGNets, ResNets and DenseNets can be recommended for use in histopathologic cancer detection, where a large dataset is available. The LeNet can be considered not suitable for the task at hand as the architecture appears not to learn in this problem space. If only a small dataset (e.g., 20,000 samples) is available, we recommend using a DenseNet, e.g., the DenseNet201. Even if it shows worse results compared to the full dataset, it is the most robust model in our evaluation. We observe that the depth of the models appears not to have a significant impact on predictive performance.

Furthermore, an early stopping (or comparable) approach should be considered using the recall or F_1 score (considering recall and precision) as the latest epoch does not always produce the best results in our evaluation. Computational effort during training can be significantly reduced that way.

The best CNN solutions proposed in the Kaggle competition achieved an AUROC score of up to 1.0 implicating perfect classification on the Kaggle test set (about 57,000 images) when training on the full training set of about 220,000 images (compared to 176,020 images for us). We did not use this test set as the labels were not available. Thus, the scores cannot be fully compared. The examined CNN-architectures (ResNet, DenseNet, and even VGGNet) already achieved AUROC scores of above 99% without explicit hyperparameter tuning. This could lead to further improvements as we did not perform it extensively due to limited resources.

As we observe very good results for all metrics with a standard classification threshold of 0.5 the influence of different thresholds on the metrics should be examined. By evaluating and comparing a variety of different metrics we are able to assess their respective suitability of the different models. Accuracy is setting true positives and true negatives equally in relation to all samples. Increasing the proportion of true positives is far more important than the number of true negatives for cancer detection. Each true positive enhances the surviving rate of patients, while false negatives could result in a larger death rate. However, more true negatives and less false positives help reduce costs for preventive medical checkups as less patients have to undertake further examinations. Lafata et al. [36] observe that false positive predictions in cancer screenings lead to cost-intensive follow-up testing. Additionally, false positive cancer screenings are associated with a psychological burden [37]. They can yield similar reductions in the quality of life compared to a true positive diagnosis [38]. Therefore, we should focus on increasing recall, while finding a low, but not equally important, false positive rate.

We can capture more true positives and decrease the number of false negatives by lowering classification thresholds. This increases the risk of false positives and reduces the true negatives at the same time. We should choose a metric that prioritizes the increase of the true positives rate (that is recall), while still taking into account false positive numbers (or precision). The AUROC metric combines recall and false positive rate. It indicates which model produces a good proportion of recall and false positive rate for different possible classification thresholds. Another appropriate metric might be the F_1 score, as it combines recall and precision. Afterall, the decision between F_1, AUROC, recall or possible other metrics to determine the best model should be subject to future work. For now, we base our model selection on the metric AUROC.

Based on our experimental results, we recommend a training set significantly larger than 20,000 samples for model training. Though, an estimation of a sufficient sample size might be evaluated in future work. We observed the AUROC metric to be very robust, while all other metrics exhibit large variance when using only a partial subset of the data (exemplary shown for AUROC and recall in Figure 7 of the Appendix).

In accordance to our expectations, we do not observe vanishing gradients, degradation or diminished information flow or any other phenomenons for VGGNets.

Despite our initial expectations, normalization and accounting for staining variances did not improve but worsen our results on almost all models. This may be caused by a multitude of reasons. Due to the black-box nature of Deep Learning models, we cannot ultimately answer whether the models focus on colors and contrasts or more on shapes. Worse results after normalization might be an

indicator of overfitting to the dataset at hand. Normalization in general might lead to better results for unseen data or other datasets. This requires further research on the original data characteristics or the effect of normalization in general. If further research yields that color or relative contrasts are taken into account, we could analyze if different normalization techniques or color deconvolution of the hematoxylin and eosin channels (as shown in the Appendix in Figure 8) and the use of Dual-Channel-RNNs might be beneficial.

With the application of normalization, we accounted for stain variances; however, other general problems as noisy ground truth labels remain. Excessive imbalanced classes, as well as the limited availability of labeled data, are not present here.

6 Conclusion

Deep Learning approaches provide the potential to make labour-intensive histopathology workflows more efficient. This paper investigated different CNN-architectures for the task of classifying malignant and non-malignant lymph node tissue on the PCam dataset. In general, all evaluated state-of-the-art CNN-architectures yielded good results with average AUROC metrics of 97-99.5%. However, a careful selection of an appropriate performance metric is essential in this medical setting and must be addressed in future works. VGGNets, ResNets, and DenseNets can be recommended for sufficiently large datasets. In the context of high medical risk, potentially achieving even slight increases might be beneficial. Tuning of hyperparameters and evaluating other Deep Learning architectures should be addressed in the future. Small datasets of fewer than 20,000 samples suffer significantly in performance. The depth of the respective network seems not to be important for the evaluated architectures. Architectures that were pretrained on ImageNet achieved equal results compared to non-pretrained. We found that stain normalization did not improve our results irrespective of the architecture, which needs to be further evaluated.

References

[1] Benjamin Barnes, Klaus Kraywinkel, Enno Nowossadeck, Ina Schönfeld, Anne Starker, Antje Wienecke, and Ute Wolf. Bericht zum krebsgeschehen in deutschland 2016. 2016.

[2] Kamila Naxerova, Johannes G Reiter, Elena Brachtel, Jochen K Lennerz, Marc Van De Wetering, Andrew Rowan, Tianxi Cai, Hans Clevers, Charles Swanton, Martin A Nowak, et al. Origins of lymphatic and distant metastases in human colorectal cancer. *Science*, 357(6346):55–60, 2017.

[3] Geert Litjens, Clara I Sánchez, Nadya Timofeeva, Meyke Hermsen, Iris Nagtegaal, Iringo Kovacs, Christina Hulsbergen-Van De Kaa, Peter Bult, Bram Van Ginneken, and Jeroen Van Der Laak. Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. *Scientific reports*, 6:26286, 2016.

[4] Jessica B O'Connell, Melinda A Maggard, and Clifford Y Ko. Colon cancer survival rates with the new american joint committee on cancer sixth edition staging. *Journal of the National Cancer Institute*, 96(19):1420–1425, 2004.

[5] Ruth Bangaoil, Abegail Santillan, Lara Mae Angeles, Lorenzo Abanilla, Antonio Lim Jr, Ma Cristina Ramos, Allan Fellizar, Leonardo Guevarra Jr, and Pia Marie Albano. Atr-ftir spectroscopy as adjunct method to the microscopic examination of hematoxylin and eosin-stained tissues in diagnosing lung cancer. *Plos one*, 15(5):e0233626, 2020.

[6] Lei He, L Rodney Long, Sameer Antani, and George R Thoma. Histology image analysis for carcinoma detection and grading. *Computer methods and programs in biomedicine*, 107(3):538–556, 2012.

[7] Sami Blom, Lassi Paavolainen, Dmitrii Bychkov, Riku Turkki, Petra Mäki-Teeri, Annabrita Hemmes, Katja Välimäki, Johan Lundin, Olli Kallioniemi, and Teijo Pellinen. Systems pathology by multiplexed immunohistochemistry and whole-slide digital image analysis. *Scientific Reports*, 7(1):1–13, 2017.

[8] Neslihan Bayramoglu, Juho Kannala, and Janne Heikkila. Deep learning for magnification independent breast cancer histopathology image classification. pages 2440–2445, 12 2016.

[9] Zhongyi Han, Benzheng Wei, Yuanjie Zheng, Yilong Yin, Kejian Li, and Shuo Li. Breast cancer multi-classification from histopathological images with structured deep learning model. *Scientific Reports*, 7, 06 2017.

[10] Nicolas Coudray, Paolo Ocampo, Theodore Sakellaropoulos, Navneet Narula, Matija Snuderl, David Fenyö, Andre Moreira, Narges Razavian, and Aristotelis Tsirigos. Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning. *Nature Medicine*, 24, 10 2018.

[11] David Steiner, Robert MacDonald, Yun Liu, Peter Truszkowski, Jason Hipp, Christopher Gammage, Florence Thng, Lily Peng, and Martin Stumpe. Impact of deep learning assistance on the histopathologic review of lymph nodes for metastatic breast cancer. *The American Journal of Surgical Pathology*, 42:1, 10 2018.

[12] Amirhossein Kiani, Bora Uyumazturk, Pranav Rajpurkar, Alex Wang, Rebecca Gao, Erik Jones, Yifan Yu, Curtis Langlotz, Robyn Ball, Thomas Montine, Brock Martin, Gerald Berry, Michael Ozawa, Florette Hazard, Ryanne Brown, Simon Chen, Mona Wood, Libby Allard, Lourdes Ylagan, and Jeanne Shen. Impact of a deep learning assistant on the histopathologic classification of liver cancer. *npj Digital Medicine*, 3, 12 2020.

[13] Miriam Hägele, Philipp Seegerer, Sebastian Lapuschkin, Michael Bockmayr, Wojciech Samek, Frederick Klauschen, Klaus-Robert Müller, and Alexander Binder. Resolving challenges in deep learning-based analyses of histopathological images using explanation methods. *Scientific reports*, 10(1):1–12, 2020.

[14] Oscar Jimenez-del Toro, Sebastian Otálora, Mats Andersson, Kristian Eurén, Martin Hedlund, Mikael Rousson, Henning Müller, and Manfredo Atzori. Analysis of histopathology images: From traditional machine learning to deep learning. In *Biomedical Texture Analysis*, pages 281–314. Elsevier, 2017.

[15] Matthew D DiFranco, Gillian O'Hurley, Elaine W Kay, R William G Watson, and Padraig Cunningham. Ensemble based system for whole-slide prostate cancer probability mapping

using color texture features. *Computerized medical imaging and graphics*, 35(7-8):629–645, 2011.

[16] Kashif Rajpoot and Nasir Rajpoot. Svm optimization for hyperspectral colon tissue cell classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 829–837. Springer, 2004.

[17] 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.

[18] F. A. Spanhol, L. S. Oliveira, C. Petitjean, and L. Heutte. Breast cancer histopathological image classification using convolutional neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2560–2567, 2016.

[19] Sabyasachi Chakraborty, Satyabrata Aich, Avinash Kumar, Sobhangi Sarkar, Jong-Seong Sim, and Hee-Cheol Kim. Detection of cancerous tissue in histopathological images using dual-channel residual convolutional neural networks (dcrcnn). In *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, pages 197–202. IEEE, 2020.

[20] Amit Kumar Jaiswal, Ivan Panshin, Dimitrij Shulkin, Nagender Aneja, and Samuel Abramov. Semi-supervised learning for cancer detection of lymph node metastases. *arXiv preprint arXiv:1906.09587*, 2019.

[21] Jingpeng Zhai, Weiran Shen, Ishwar Singh, Tom Wanyama, and Zhen Gao. A review of the evolution of deep learning architectures and comparison of their performances for histopathologic cancer detection. *Procedia Manufacturing*, 46:683–689, 2020.

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

[23] M. Macenko, M. Niethammer, J. S. Marron, D. Borland, J. T. Woosley, Xiaojun Guan, C. Schmitt, and N. E. Thomas. A method for normalizing histology slides for quantitative analysis. In *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1107–1110, 2009.

[24] Katharina Glatz, Udo Spornitz, Alain Spatz, Michael Mihatsch, and Dieter Glatz. Factors to keep in mind when introducing virtual microscopy. *Virchows Archiv : an international journal of pathology*, 448:248–55, 04 2006.

[25] A Ljungberg and O Johansson. Methodological aspects on immunohistochemistry in dermatology with special reference to neuronal markers. *The Histochemical journal*, 25(10):735—745, October 1993.

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[27] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[28] Wenlong Li, Xingguang Li, Yueya Qin, Wenjun Song, and Wei Cui. Application of improved lenet-5 network in traffic sign recognition. In *Proceedings of the 3rd International Conference on Video and Image Processing*, ICVIP 2019, page 13–18, New York, NY, USA, 2019. Association for Computing Machinery.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[32] 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*, pages 4700–4708, 2017.

[33] Marian Tietz, Thomas J. Fan, Daniel Nouri, Benjamin Bossan, and skorch Developers. *skorch: A scikit-learn compatible neural network library that wraps PyTorch*, July 2017.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[36] Jennifer Elston Lafata, Janine Simpkins, Lois Lamerato, Laila Poisson, George Divine, and Christine Cole Johnson. The economic impact of false-positive cancer screens. *Cancer Epidemiology and Prevention Biomarkers*, 13(12):2126–2132, 2004.

[37] Patricia M McGovern, Cynthia R Gross, Richard A Krueger, Deborah A Engelhard, Jill E Cordes, and Timothy R Church. False-positive cancer screens and health-related quality of life. *Cancer Nursing*, 27(5):347–352, 2004.

[38] John Brodersen and Volkert Dirk Siersma. Long-term psychosocial consequences of false-positive screening mammography. *The Annals of Family Medicine*, 11(2):106–115, 2013.

Appendix

	pretrained				non-pretrained			
	normalized		non-normalized		normalized		non-normalized	
	full	partial	full	partial	full	partial	full	partial
Accuracy								
DenseNet	-	-	94.55	-	95.90	-	97.20	89.93
VGGNet	-	-	-	-	95.97	-	96.67	86.64
ResNet	-	-	97.11	-	94.55	-	96.07	86.97
F_1 Score								
DenseNet	-	-	93.20	-	94.96	-	96.54	86.91
VGGNet	-	-	-	-	95.00	-	95.86	83.32
ResNet	-	-	96.43	-	93.37	-	95.15	83.23
Precision								
DenseNet	-	-	94.18	-	95.19	-	96.52	91.02
VGGNet	-	-	-	-	96.11	-	96.21	83.69
ResNet	-	-	96.45	-	92.68	-	95.03	85.91
Recall								
DenseNet	-	-	92.24	-	94.74	-	96.56	83.44
VGGNet	-	-	-	-	93.91	-	95.52	83.10
ResNet	-	-	96.41	-	94.10	-	95.32	81.21
AUROC								
DenseNet	-	-	94.18	-	97.18	-	97.90	96.46
VGGNet	-	-	-	-	97.41	-	97.36	93.95
ResNet	-	-	99.53	-	96.66	-	97.09	94.12

Table 2: Architecture comparison (in %, weighted mean, classification threshold = 0.5)

	pretrained				non-pretrained			
	normalized		non-normalized		normalized		non-normalized	
	full	partial	full	partial	full	partial	full	partial
Accuracy								
mean	-	-	-	-	95.97	-	96.67	86.64
min	-	-	-	-	95.73	-	93.64	86.33
max	-	-	-	-	96.21	-	97.48	87.00
F_1 Score								
mean	-	-	-	-	95.00	-	95.86	83.32
min	-	-	-	-	94.72	-	92.10	82.47
max	-	-	-	-	95.27	-	96.88	83.91
Precision								
mean	-	-	-	-	96.11	-	96.21	83.69
min	-	-	-	-	95.36	-	92.54	80.14
max	-	-	-	-	96.47	-	97.15	85.06
Recall								
mean	-	-	-	-	93.91	-	95.52	83.10
min	-	-	-	-	93.03	-	89.74	80.04
max	-	-	-	-	94.27	-	96.80	88.06
AUROC								
mean	-	-	-	-	97.41	-	97.36	93.95
min	-	-	-	-	95.87	-	93.32	93.72
max	-	-	-	-	98.95	-	99.48	94.25

Table 3: VGGNet architecture (in %, weighted mean, classification threshold = 0.5)

	pretrained				non-pretrained			
	normalized		non-normalized		normalized		non-normalized	
	full	partial	full	partial	full	partial	full	partial
Accuracy								
mean	-	-	97.11	-	94.55	-	96.07	86.97
min	-	-	97.11	-	93.83	-	93.49	83.88
max	-	-	97.11	-	95.39	-	97.19	90.58
F_1 Score								
mean	-	-	96.43	-	93.37	-	95.15	83.23
min	-	-	96.43	-	92.53	-	91.67	79.30
max	-	-	96.43	-	94.14	-	96.53	88.07
Precision								
mean	-	-	96.45	-	92.68	-	95.03	85.91
min	-	-	96.45	-	90.72	-	89.09	81.28
max	-	-	96.45	-	95.40	-	96.95	90.58
Recall								
mean	-	-	96.41	-	94.10	-	95.32	81.21
min	-	-	96.41	-	92.70	-	88.55	70.52
max	-	-	96.41	-	96.30	-	96.86	89.93
AUROC								
mean	-	-	99.53	-	96.66	-	97.09	94.12
min	-	-	99.53	-	93.89	-	92.70	90.58
max	-	-	99.53	-	98.73	-	99.39	96.41

Table 4: ResNet architecture (in %, weighted mean, classification threshold = 0.5)

	pretrained				non-pretrained			
	normalized		non-normalized		normalized		non-normalized	
	full	partial	full	partial	full	partial	full	partial
Accuracy								
mean	-	-	94.55	-	95.90	-	97.20	89.93
min	-	-	94.22	-	95.64	-	96.60	89.10
max	-	-	94.88	-	96.24	-	97.65	90.67
F_1 Score								
mean	-	-	93.20	-	94.96	-	96.54	86.91
min	-	-	92.82	-	94.70	-	95.75	85.77
max	-	-	93.58	-	95.31	-	97.09	88.70
Precision								
mean	-	-	94.18	-	95.19	-	96.52	91.02
min	-	-	93.31	-	94.35	-	94.83	86.47
max	-	-	95.05	-	96.34	-	97.47	94.74
Recall								
mean	-	-	92.24	-	94.74	-	96.56	83.44
min	-	-	92.15	-	94.31	-	94.81	78.36
max	-	-	92.33	-	95.05	-	97.57	91.04
AUROC								
mean	-	-	94.18	-	97.18	-	97.90	96.46
min	-	-	93.92	-	95.75	-	96.31	95.68
max	-	-	94.45	-	99.08	-	99.51	96.73

Table 5: DenseNet architecture (in %, weighted mean, classification threshold = 0.5)

	ResNet152	ResNet152
Epochs	30	30
Learning Rate	0.01	0.01
Avg Duration/Epoch in s	408	426
Full Data	full	full
Normalized	no	no
Optimizer	Adam	Adam
Pretrained	no	no
Test Accuracy	93.49	93.93
Test F_1	91.67	92.81
Test Precision	95.02	89.09
Test Recall	88.55	96.86
Test AUROC	92.70	94.41

Table 6: Worst test metrics (in %)

Architecture	Pretrained	Normalized	Full data	Accuracy	F_1	Precision	Recall	AUROC
DenseNet121	no	yes	yes	95.97	95.02	95.53	94.53	96.92
DenseNet121	no	no	yes	97.23	96.58	96.46	96.70	97.73
DenseNet121	no	no	no	90.28	87.58	90.13	85.51	96.56
DenseNet121	yes	no	yes	94.22	92.82	93.31	92.33	93.92
DenseNet201	no	yes	yes	95.84	94.90	94.85	94.95	97.44
DenseNet201	no	no	yes	97.17	96.51	96.59	96.43	98.06
DenseNet201	no	no	no	89.57	86.25	91.90	81.36	96.37
DenseNet201	yes	no	yes	94.88	93.58	95.05	92.15	94.45
ResNet152	no	yes	yes	94.30	93.13	91.43	94.91	96.58
ResNet152	no	no	yes	95.29	94.20	93.93	94.56	96.35
ResNet152	no	no	no	84.84	79.79	86.26	74.46	92.44
ResNet18	no	yes	yes	94.80	93.61	93.93	93.29	96.75
ResNet18	no	no	yes	96.85	96.11	96.13	96.09	97.83
ResNet18	no	no	no	89.10	86.68	85.57	87.96	95.79
ResNet34	yes	no	yes	97.11	96.43	96.45	96.41	99.53
VGG11	no	yes	yes	95.96	95.00	95.79	94.23	97.41
VGG11	no	no	yes	96.06	95.10	95.57	94.63	96.58
VGG11	no	no	no	86.71	83.78	82.35	85.42	94.09
VGG19	no	yes	yes	95.98	94.99	96.44	93.59	97.41
VGG19	no	no	yes	97.27	96.62	96.85	96.40	98.15
VGG19	no	no	no	86.56	82.86	85.04	80.78	93.82

Table 7: Comparison of all different configurations of network architectures (metrics in %)

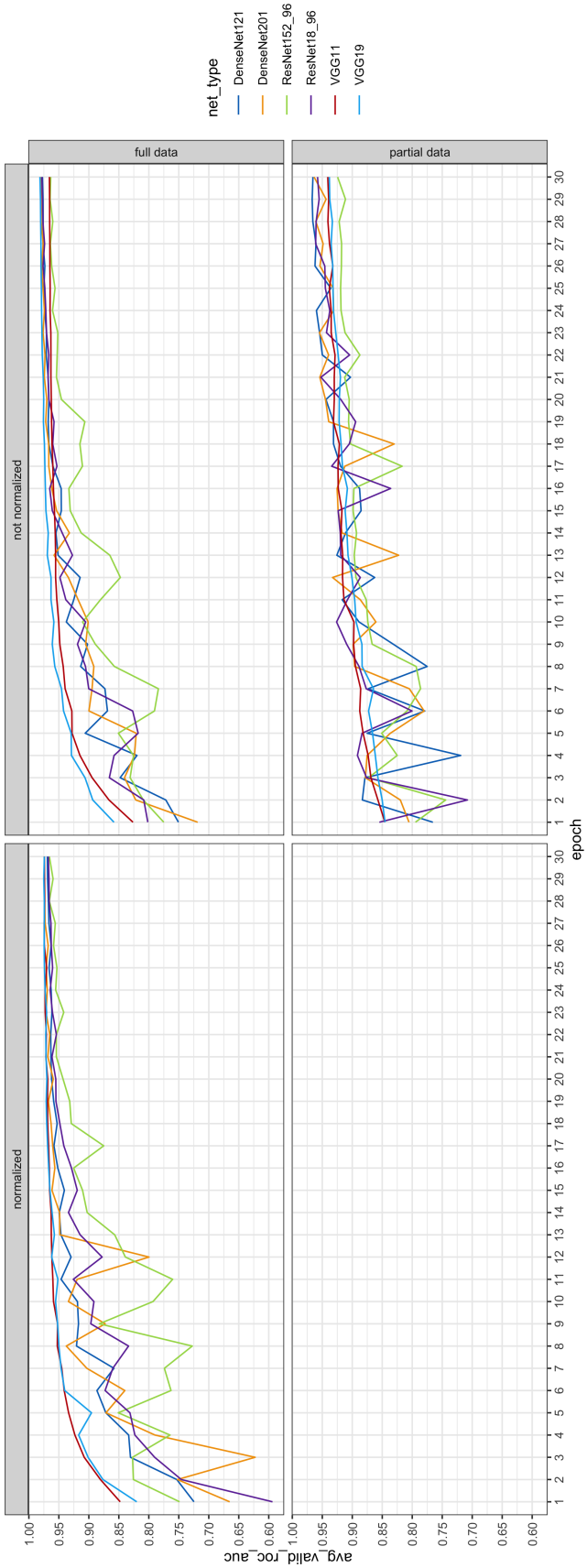


Figure 4: AUROC plot for non-pretrained runs per model architecture

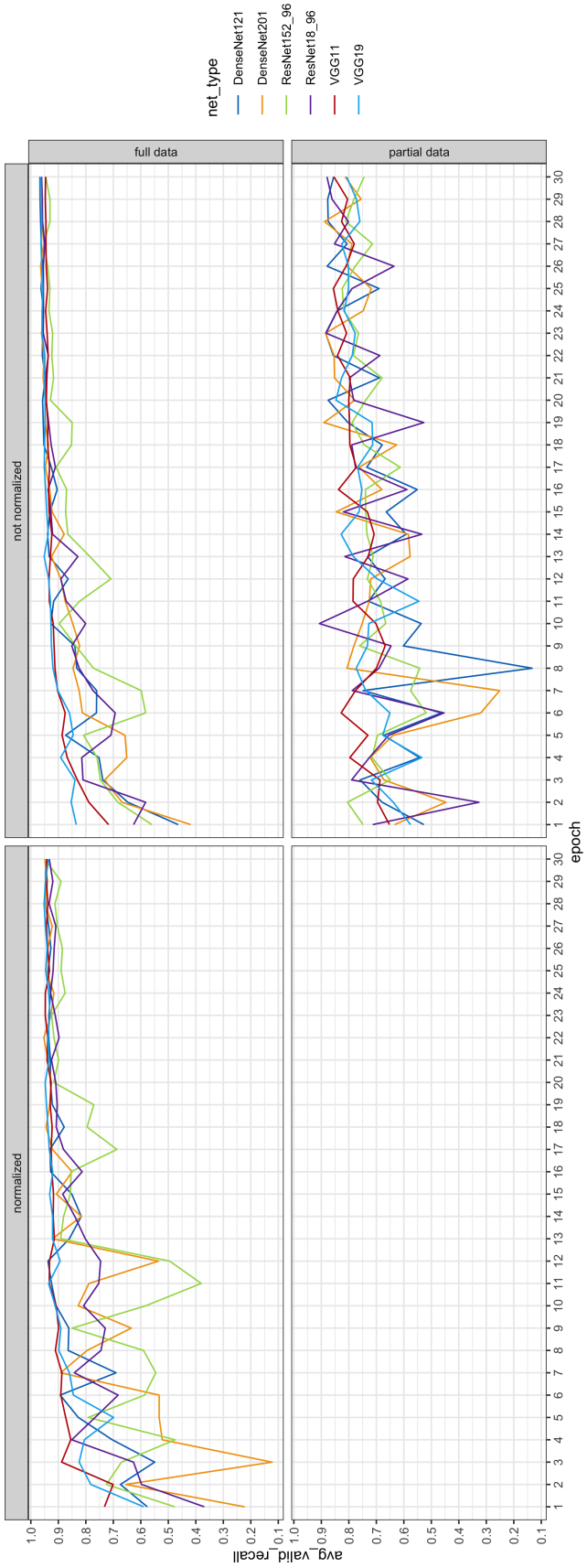


Figure 5: Recall plot for non-pretrained runs per model architecture

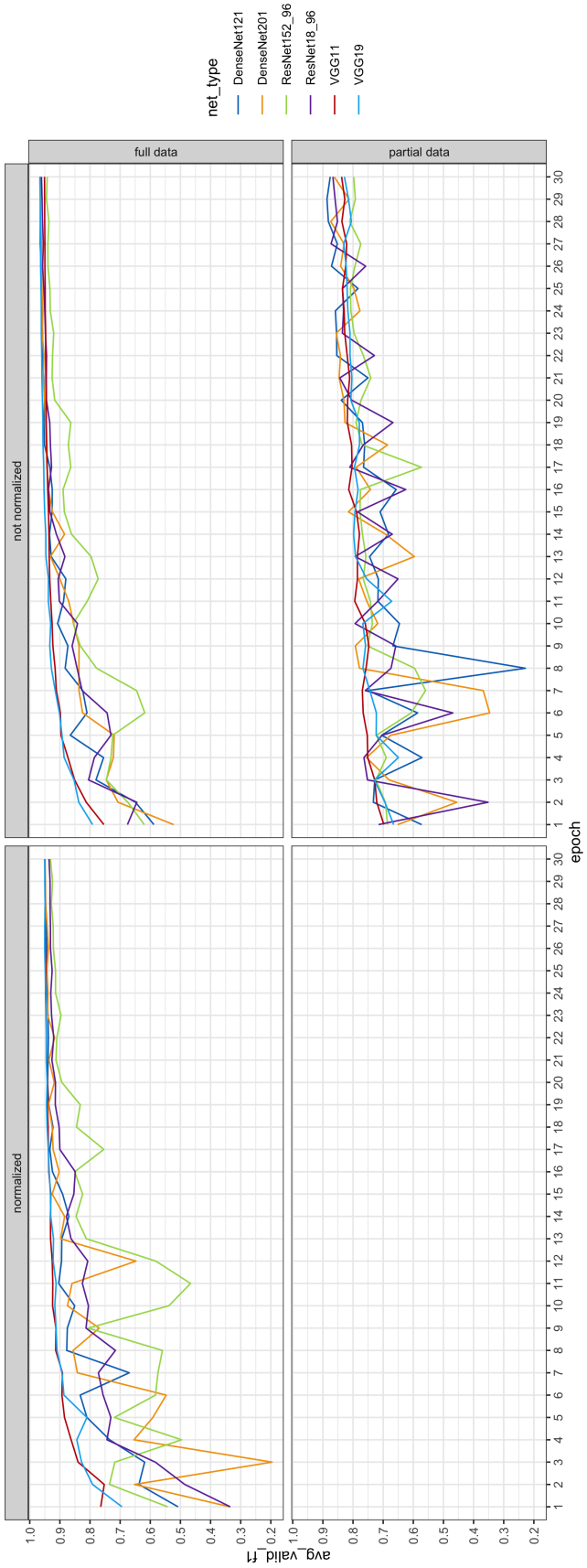


Figure 6: F_1 plot for non-pretrained runs per model architecture

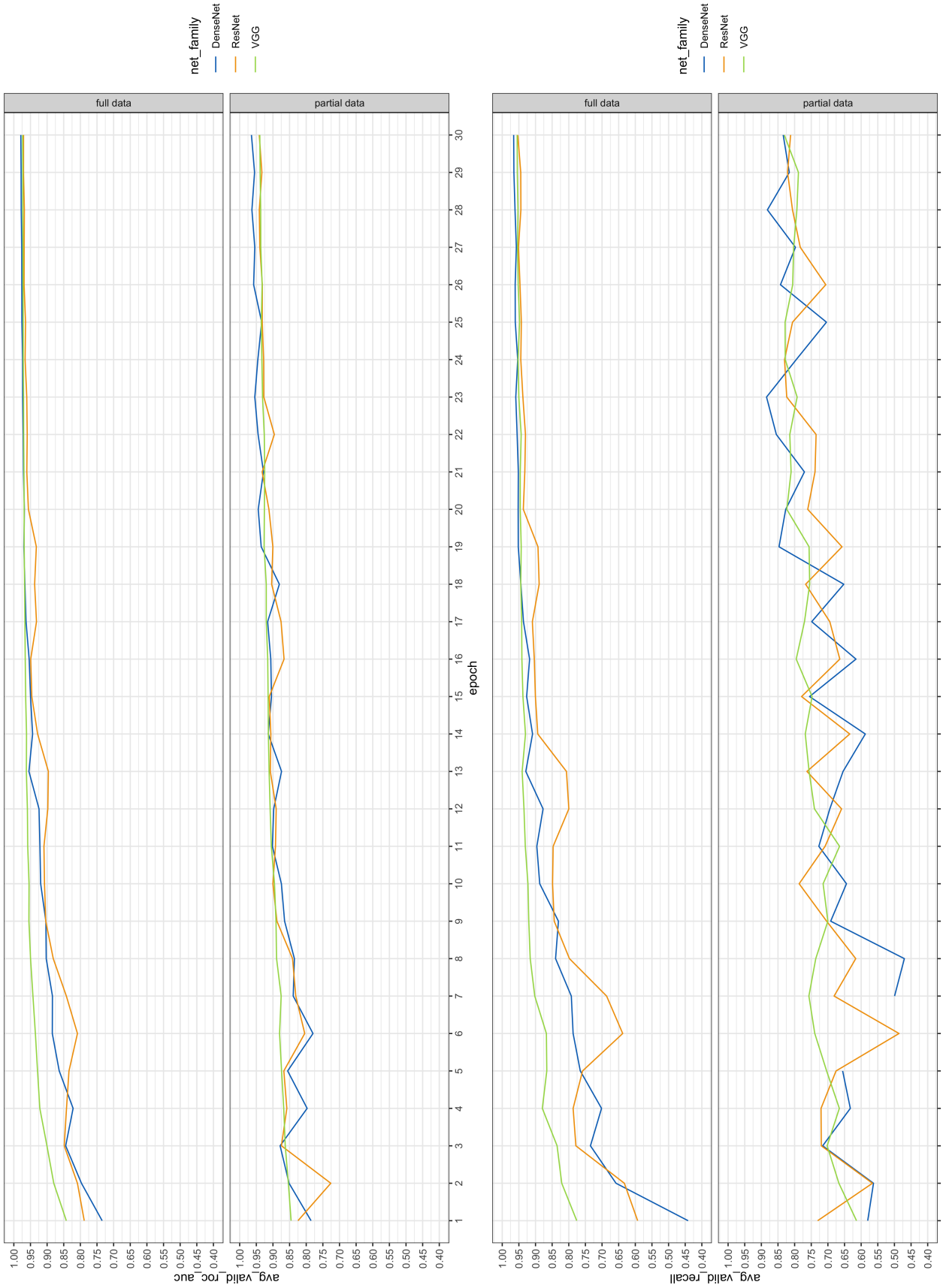


Figure 7: Weighted mean AUROC plot (top) and average recall (bottom) plot for full and partial data each.

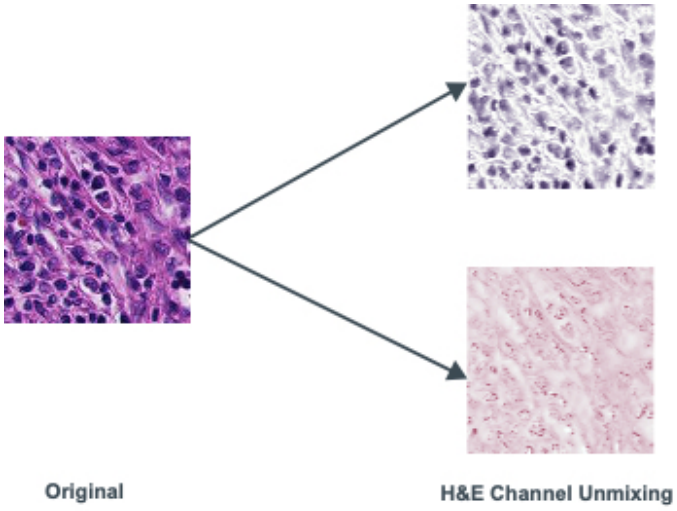


Figure 8: Splitting into H&E channels during normalization