Automated Architecture-Modeling for Convolutional Neural Networks

Manh Khoi Duong[1]

Abstract: Tuning hyperparameters can be very counterintuitive and misleading, yet it plays a big (or even the biggest) part in many machine learning algorithms. For instance, finding the right architecture for an artificial neural network (ANN) can also be seen as a hyperparameter e.g. number of convolutional layers, number of fully connected layers etc. Tuning these can be done manually or by techniques such as grid search or random search. Even then finding optimal hyperparameters seems to be impossible. This paper tries to counter this problem by using bayesian optimization, which finds optimal parameters, including the right architecture for ANNs. In our case, a histological image dataset was used to classify breast cancer into stages.

Keywords: CNN; Model Architecture; Breast Cancer; Histology

1 Introduction

Hyperparameters in machine learning are properties of an algorithm. There exist many hyperparameters and especially neural networks have numerous of them. An essential task is to find the hyperparameters which lead to the best results of the proposed classifier. Techniques such as *grid search* and *random search* [BB12] can be proposed to solve this task but lack of the ability to learn from previous outcomes. This is where *Bayesian optimization* [Mo75] comes into play. This paper describes how Bayesian optimization can be used to counter the problem of finding the right model architecture for Convolutional Neural Networks to yield for the best results. The final evaluation of the proposed method in this paper was done on a histological dataset [Ar17].

2 Related works

Similar works whose task is to optimize the model architecture are [ZL16], [Li17] and [Zo17]. The mentioned works use *reinforcement learning* to consider the results of the previous architectures to find better ones after each step. A *controller*, which is equivalent to an agent, trains a *child network* each step. The feedback scores for the controller are the

[1] Heinrich-Heine-Universität, Datenbanken und Informationssysteme, Universitätsstraße 1, 40225 Düsseldorf, Germany manh.duong@hhu.de

doi:10.18420/btw2019-ws-17

validation accuracies. This guides the controller to produce better models at each iteration. In contrast to this paper, the methods in [ZL16], [Li17] and [Zo17] use recurrent neural networks where hyperparameters have to be set, too. Differently, the methods used in this paper require only two hyperparameters which are the number of models to be trained and the choice of the *acquisition function*. The trade-off seems to be better as fewer settings have to be considered to optimize a function which requires much more settings.

3 Foundation

Bayesian optimization is a global optimization algorithm and works by firstly making a *prior belief* about the *objective function*[2]. Then a prediction is made to find the minimum or maximum. After observing the current evidence, the *posterior belief* is updated by the given evidence and the prior belief. This update step is repeated for a defined number of iterations so that the uncertainty decreases and the objective function can be approximated. Assuming that each point of the function is normally distributed, then this function can be modeled with *Gaussian process (GP)* regression. It can target multiple variables, thus multiple hyperparameters can be optimized.
To determine the next point x_{next} for an observation X, an acquisition function is required. This function usually noted by $a : X \rightarrow \mathbb{R}^+$, with set of points X, takes all observations and the best result into account [BCdF10]. It also has to be maximized to select the next point x_{next} to evaluate the function at e.g. $x_{next} = argmax_x a(x)$. Solving another optimization problem than the objective function seems to be impractical but the acquisition function is usually easier to optimize and much cheaper. It also limits the regression task of the objective function: not all points are going to be evaluated, only points which are important to find the minimum or maximum. An acquisition deals with the trade-off between *exploration* and *exploitation*. Exploration is discovering regions with higher uncertainty and exploitation is sampling on regions that will likely offer an improvement [BCdF10]. Exploration is needed to escape from the local optimum.

Consider the hyperparameter optimization task with n hyperparameters, the objective function is the function that samples all hyperparameter settings. Each point is n-dimensional and contains values for each hyperparameter. The objective function has to be maximized because a high accuracy is yielded. This can be formalized as $argmax_x f(\vec{x}, D, y, D_{val}, y_{val})$ where x is the n-dimensional vector containing hyperparameter values, D is the training data, D_{val} is the validation data, y and y_{val} are labels with f being a classifier which returns the validation accuracy. The search space can be defined as intervals, making it continuous as in random search.

[2] A function to minimize or maximize

4 Methods

This chapter aims to explain the method of how hand tuning has been replaced. The proposed method can be divided into two steps: finding the model and fine-tuning the model. The first subsection deals with the aspect which models can be found and the second subsection deals with the details of fine-tuning the best model that has been found. This method has been split in two because searching for all hyperparameters at once is not practical considering the exponential growth of the search space.

4.1 Model finding

As already said in Section 3, the classifier can be seen as a function f which takes $\vec{x}, D, y, D_{val}, y_{val}$ as inputs and returns the validation accuracy. Due to the fact that the used python package [Sc] minimizes the objective, the optimization problem is $argmin_x \widetilde{f}(\vec{x}, D, y, D_{val}, y_{val})$ with $\widetilde{f}$ being the classifier which returns the negative validation accuracy. Note that the goal is to find $\vec{x}$.

The objective is called by the bayesian optimization algorithm which can be seen in Algorithm 1 [BCdF10]. The attended hyperparameter $\vec{x}$ and its linked validation_accuracy = $F(\vec{x}, D, y, D_{val}, y_{val})$ are being saved to $\widetilde{D}$ which is not to be confused with the training or validation data. The Gaussian process regression is then updated in the last step. The best hyperparameters can then be observed after n iteration steps of calling the objective with different $\vec{x}$. This optimization method does not have many hyperparameters itself. Only the number of calls and the acquisition function is needed. For this paper, Expected Improvement was chosen as the acquisition function and is defined for the minimization task as follows [JSW98]:

$$EI(x) = \mathbb{E}[max\,\{0, f(\widetilde{x}) - f(x)\}]$$

where $\widetilde{x}$ is the best hyperparameter that has been observed so far. This formula expresses that the current $f(x)$ has to be smaller than $f(\widetilde{x})$ and their difference should be maximized. The expected improvement can then be obtained by the expected value.

The model architectures proposed by this algorithm follow the trend of how modern convolutional neural networks are built (see comparison with VGG16, VGG19 [SZ14], DenseNet [Hu17]). The created model architectures are sequential and nonrecurrent. As seen in Figure 1, the first convolutional block can be of size m, meaning it can contain m convolutional (Conv) layers activated with the ReLu function. The second Conv block can be of size n. The batch normalization (BN) layer was marked differently in the Figure as the use of it was set to be a hyperparameter. Convolutional layers that are contained in the same block have equivalent output dimensions and filter sizes. After several additional p Conv blocks, the output is fed to k dense layers. A dropout layer is added between each dense layer. The dense layers were also activated with the ReLu function. The last layer is

Algorithm 1 Bayesian Optimization

procedure BAYESOPT($\vec{x}, D, y, D_{val}, y_{val}$)
  for $i \leftarrow 1, n$ **do**
    $x_{\vec{next}} \leftarrow argmax_{\vec{x}} acq(\vec{x}, \widetilde{D})$ ▷ acq is the acquisition function
    val_accuracy $\leftarrow \tilde{f}(x_{\vec{next}}, D, y, D_{val}, y_{val})$
    if val_accuracy < best_accuracy **then**
      best_accuracy $\leftarrow$ val_accuracy
      $x_{\vec{best}} \leftarrow x_{\vec{next}}$
    end if
    $\widetilde{D} \leftarrow \widetilde{D} \cup (x_{\vec{next}}$, val_accuracy)
    Update Gaussian process regression based on $\widetilde{D}$
  end for
  return $x_{\vec{best}}$
end procedure

the softmax function which produces probabilities for the final output. Note that after each Conv Block, a maxpooling (MP) layer is added consecutively. The MP layer reduces the shape it receives into a much smaller shape. Too many MP layers can lead to an exception of negative output shapes. The maximum amount of Conv blocks is therefore limited by the very first input shape. Larger image sizes can have deeper models because more MP layers are allowed. The algorithm developed for this paper is able to adapt the maximum amount of Conv blocks to be created based on the input shape. To prevent underfitting, the parameters m, n, p (see Figure 1) are at least one. Differently, it is possible that the created neural network can have zero additional dense layers ($k = 0$).

4.2 Fine-tuning

The fine-tuning was applied on the model that has achieved the best results on the model finding algorithm. The algorithm in fine-tuning is the same as described in Algorithm 1. The only difference is that the objective gets different hyperparameters. In the fine-tuning process a total of eight hyperparameters were tuned. The output dimension of the first, second and other Conv blocks make up three hyperparameters. The rest consists of the learning rate, dropout, number of dense nodes, lambda value of the L2-regularization and the kernel size of each Conv filter. All other settings that are not mentioned were preserved from the model finding task e.g. the optimizer and activation function were not changed. The hyperparameters which are to be optimized in this task can be easily switched up with other hyperparameters available. The hyperparameter vector $\vec{x}$ and the search space could be defined arbitrary.

After finding the right model and fine-tuning the model, it was aimed to train the model further to achieve even better results.

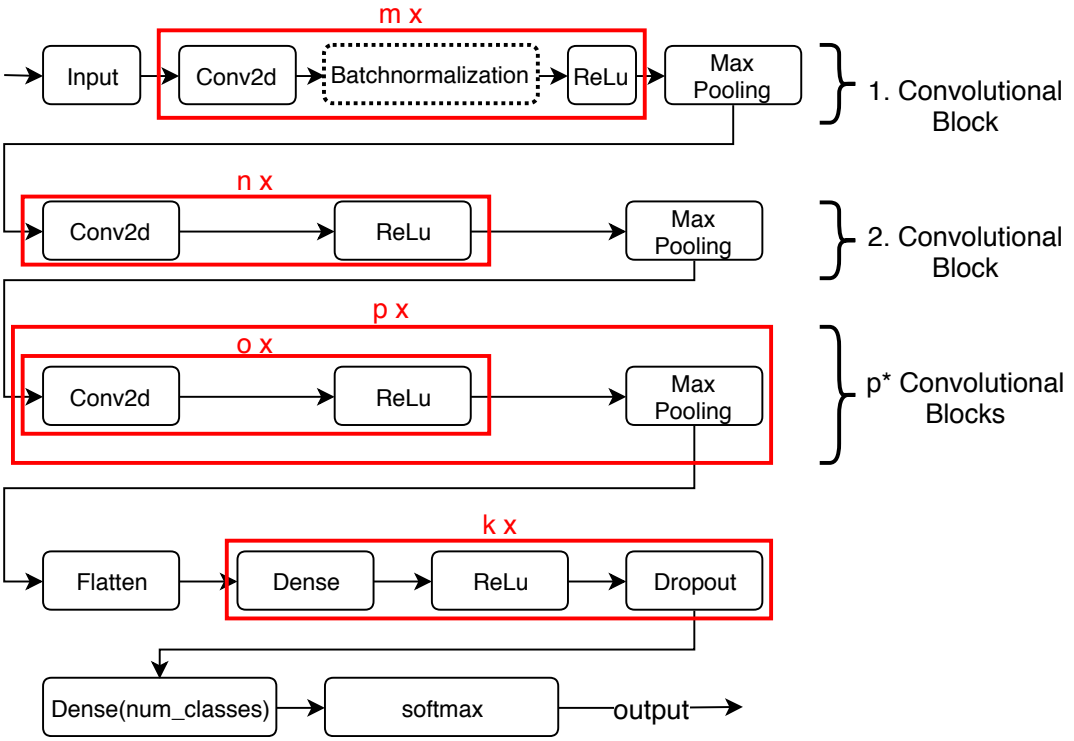
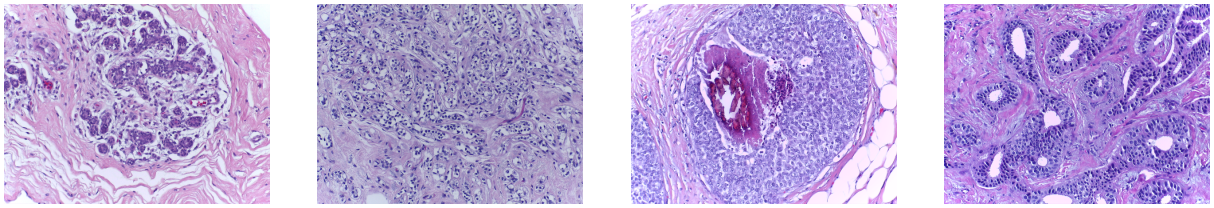


Fig. 1: Model architectures that can possibly be created

5 Evaluation

The application of the methods was done on a histological dataset of breast cancer (see Figure 2). The dataset provided by [Ar17] consists of 249 training samples and 36 test samples labeled in the following four classes: *normal, benign, in situ carcinoma* and *invasive carcinoma*. This leads to a four-class classification task. Another task that comes up by this dataset is the classification into *carcinoma* and *noncarcinoma* cells where *normal, benign* are noncarcinoma and *in situ, invasive* are carcinoma. Each image has a resolution of 2048×1536 pixels. The breast tissues were stained with haematoxylin and eosin (H&E) and were digitized under the same conditions. The magnification on this tissue is 200×. The provided test set contains 20 images where the images can be classified clearly (labeled as *initial* in [Ar17]) and 16 additional images with increased ambiguity. The latter were labeled as *extended*. The classes have almost evenly distributed samples.
Because neural networks require a high amount of training samples, the dataset was augmented with flippings, mirrorings, rotations and random contrast changes. The augmentation increased the number of images by eight times. This results in 1992 samples to train the neural network.



(a) Normal (b) Benign (c) In Situ (d) Invasive

Fig. 2: Samples of each stage

5.1 Inspecting bayesian optimization

Due to the use of bayesian optimization to find the optimal hyperparameters, its accomplishment in this task can be studied further in this section. From a theoretical point of view, this optimization algorithm should find better values after each evaluation of the objective function and a convergence should be seen. To see how this optimization technique performed, a plot (see Figure 3) has been made which shows the accuracy of each created model in the y-axis and the nth model in the x-axis. It was set that each created model has to be trained for 40 epochs. Note that the fine tuning process was not done yet in this figure and only the created models are considered.

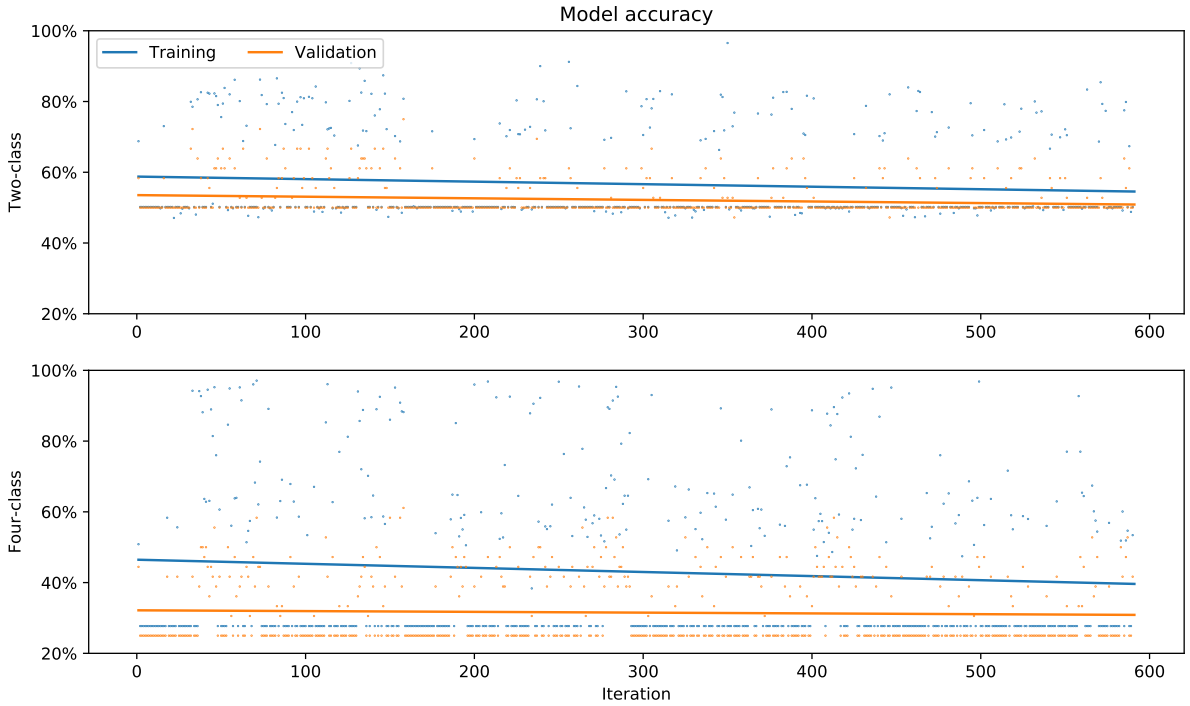


Fig. 3: Accuracy of the created models on both classes. A linear regression for each validation and training accuracy describes how bayesian optimization finds optimal models.

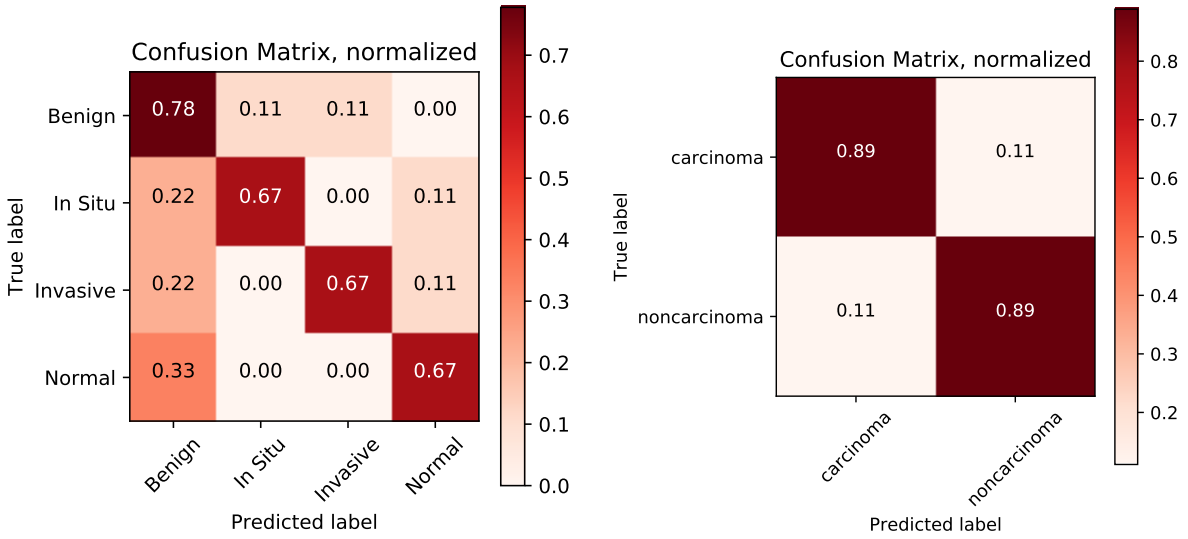
Considering the two-class classification, the values in the y-axis ranges from slightly under 50% to 100%. This is because guessing in a binary classification problem will already

result in an accuracy of 50% on average if the dataset is evenly distributed which is fulfilled in our given dataset. A lot of models that were created, scored bad accuracies. Meaning those models were not able to learn from the data. This can be seen in the figure where a lot of points lie in the accuracy of 50% which nearly creates a horizontal line visually. The same can be observed in the four-class classification where the threshold is 25%. If the plot as a whole is considered, there is no evidence of a convergence obtainable. All points appear to be randomly distributed. The linear regression even shows a decrease in both training and validation accuracy with each model that is found. This contradicts with the theoretical assumption that better models should be found. For both classification tasks, the best validation accuracy was found after 157 iterations. The achieved validation accuracy in the four-class task is 61%. For the two-class task, the achieved validation accuracy is 75%. This might be caused by randomly bad weight initializations or by the exploration step that was done too much.

5.2 Final results

The best model that was found as described in Section 5.1 was fine-tuned and trained for additional epochs. 1000 different hyperparameter settings were tested and it was chosen to let the model train for 60 epochs with each hyperparameter setting. The results which are presented in the following are based on the validation set: The confusion matrices in Figure 4 are normalized which means that the number of predictions were divided by the total number of images per class. This results in class-wise recall scores on the diagonal. The confusion matrix in Figure 4a) shows on the diagonal that nearly all classes were predicted correctly with a probability of 67%. Only benign has a recall score of 78%. It can be misleading to state that benign tissues get classified at best because the first column shows that benign was predicted at most and therefore the ANN was already likely to score best at it. The higher number of predictions on the benign class can be explained by the slightly unbalanced class distribution as it makes up 28% of the training data. The precision score for the benign class is only 50%. Because normal and benign belong to non-carcinoma cells, the most confusion happened there: $\frac{1}{3}$ of the normal tissues were labeled as benign but the opposite did not happen. The confusion matrix in 4b) shows that the classifier stages carcinoma and non-carcinoma tissue both with a probability of 89%. As seen, the false positive rate and false negative rate is 11%.

Table 1 shows different metrics to evaluate the classifiers. As seen, the metrics score for two classes is the same everywhere. This is due to the equally distributed dataset and - as seen in Figure 4b) - that all classes were fortunately predicted right with the same probability. For non-binary classes there exists different averaging methods for the measures. The chosen averaging method is *macro* which computes the given measure for each class and then takes the arithmetic mean of it. As seen, the precision score is higher than the recall score for four classes. The neural network is therefore good at classifying into the right classes but misses a few elements which belong to it.



(a) Four-class (b) Two-class

Fig. 4: Confusion matrices of the ANN on both classification tasks

Classes	Accuracy	Recall	Precision	F1-score
Two-class	89%	89%	89%	89%
Four-class	69%	69%	74%	70%

Tab. 1: Different metrics to evaluate the ANNs

5.3 Comparing with related works

The following comparison is done against [GAS18], [Na18] and [NAE18] who made their papers publicly available. The mentioned works participated in the *ICIAR 2018 Grand Challenge on Breast Cancer Histology Images* [Ar18]. The dataset provided in this challenge is based on the dataset [Ar17] used in this paper but consists of 151 more samples which is an advantage. The dataset provided by the challenge was not used here because the dataset was only made available for the participants. The comparison can be seen in Table 2 and all results are based on the validation accuracy.

Team	4-class acc.	2-class acc.	Approach
Aditya et al. [GAS18]	85%	93%	Transfer learning: Inception-v3 [Sz15]
Wajahat et al. [Na18]	81%	-	Transfer learning: AlexNet [KSH12]
Kamyar et al. [NAE18]	95%	-	Transfer learning: Inception-v3 [Sz15]
Own work	**69%**	**89%**	**Automated Architecture-Modeling**

Tab. 2: Validation accuracies compared with similar works

As seen in Table 2, not every team did the two-class classification. The comparison for two classes can only be done against Aditya et al. [GAS18]. It can be observed that the results are very similar in this case. For four classes, this work achieved inferior results against the participants of the challenge. This can be caused by the higher amount of samples the

teams had or by the use of a very common method of *transfer learning*. Though transfer learning has been used by every mentioned team, the results can still differ. The results can even vary if the teams use the same base model ([NAE18] and [KSH12]). As the number of samples is essential for neural networks, the differences in the results can likewise be caused by the different preprocessing and data augmentation methods. Apparently the validation set is not the same for every team which can also lead to an inaccurate comparison. [GAS18] and [Na18] used a fixed validation set throughout the hyperparameter search whereas [NAE18] used cross-validation. The split was set differently by every participant. The results, therefore, should be obtained with caution.

6 Conclusion and future works

Though the presented methods were applied on only one dataset consisting of two tasks, the results were quite decent as an accuracy of 89% was achieved in the binary classification task. For the four-class task, an accuracy of 69% was achieved. Also, it has been shown that the real application of bayesian optimization neglects the intuitive results as no convergence could be observed. This can be caused by many reasons, a further inspection and application of the methods could be done in future works.

Furthermore, the model finding method could be extended by allowing the creation of recurrent and nonsequential models. Considering more hyperparameters could lead to a closer goal to the global optimum. Though the problem might be the curse of dimensionality when considering too many possibilities, a bypass is to split the task into more steps instead of two. The model finding algorithm could also be optimized: Some models already lack to learn from the data and further training is not required. Some models, however, can accomplish better results when being trained on a very high amount of epochs. When considering the number of epochs as the depth and the amount of models that are going to be created as the breadth, an application of *iterative deepening depth-first search* which combines the *breadth-first* and *depth-first* search can thus be applied.

References

[Ar17] Araújo, Teresa; Aresta, Guilherme; Castro, Eduardo; Rouco, José; Aguiar, Paulo; Eloy, Catarina; Polónia, António; Campilho, Aurélio; et al: Classification of breast cancer histology images using Convolutional Neural Networks. PLOS ONE, 12(6), 2017.

[Ar18] Aresta, Guilherme; Araújo, Teresa; Kwok, Scotty; Chennamsetty, Sai Saketh; P., Mohammed Safwan K.; Varghese, Alex; Marami, Bahram; Prastawa, Marcel; Chan, Monica; Donovan, Michael J.; Fernandez, Gerardo; Zeineh, Jack; Kohl, Matthias; Walz, Christoph; Ludwig, Florian; Braunewell, Stefan; Baust, Maximilian; Vu, Quoc Dang; To, Minh Nguyen Nhat; Kim, Eal; Kwak, Jin Tae; Galal, Sameh; Sanchez-Freire, Veronica; Brancati, Nadia; Frucci, Maria; Riccio, Daniel; Wang, Yaqi; Sun, Lingling; Ma, Kaiqiang; Fang, Jiannan; Koné, Ismaël; Boulmane, Lahsen; Campilho, Aurélio; Eloy, Catarina; Polónia, António; Aguiar, Paulo: BACH: Grand Challenge on Breast Cancer Histology Images. CoRR, abs/1808.04277, 2018.

[BB12] Bergstra, James; Bengio, Yoshua: Random Search for Hyper-parameter Optimization. J. Mach. Learn. Res., 13:281–305, 2012.

[BCdF10] Brochu, Eric; Cora, Vlad M.; de Freitas, Nando: , A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning, 2010.

[GAS18] Golatkar, Aditya; Anand, Deepak; Sethi, Amit: Classification of Breast Cancer Histology using Deep Learning. CoRR, abs/1802.08080, 2018.

[Hu17] Huang, G.; Liu, Z.; v. d. Maaten, L.; Weinberger, K. Q.: Densely Connected Convolutional Networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2261–2269, 2017.

[JSW98] Jones, Donald R.; Schonlau, Matthias; Welch, William J.: Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization, 13:455–492, 1998.

[KSH12] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E: ImageNet Classification with Deep Convolutional Neural Networks. In (Pereira, F.; Burges, C. J. C.; Bottou, L.; Weinberger, K. Q., eds): Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc., 2012.

[Li17] Liu, Chenxi; Zoph, Barret; Shlens, Jonathon; Hua, Wei; Li, Li-Jia; Fei-Fei, Li; Yuille, Alan L.; Huang, Jonathan; Murphy, Kevin: Progressive Neural Architecture Search. CoRR, abs/1712.00559, 2017.

[Mo75] Močkus, J.: On bayesian methods for seeking the extremum. Springer Berlin Heidelberg, pp. 400–404, 1975.

[Na18] Nawaz, Wajahat; Ahmed, Sagheer; Tahir, Muhammad; Khan, Hassan: Classification Of Breast Cancer Histology Images Using ALEXNET. pp. 869–876, 06 2018.

[NAE18] Nazeri, Kamyar; Aminpour, Azad; Ebrahimi, Mehran: Two-Stage Convolutional Neural Network for Breast Cancer Histology Image Classification. In: International Conference Image Analysis and Recognition. Springer, pp. 717–726, 2018.

[Sc] Scikit-optimize. Website. Online available: `https://github.com/scikit-optimize/scikit-optimize`; visited 29/07/2018.

[SZ14] Simonyan, Karen; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556, 2014.

[Sz15] Szegedy, Christian; Vanhoucke, Vincent; Ioffe, Sergey; Shlens, Jonathon; Wojna, Zbigniew: Rethinking the Inception Architecture for Computer Vision. CoRR, abs/1512.00567, 2015.

[ZL16] Zoph, Barret; Le, Quoc V.: Neural Architecture Search with Reinforcement Learning. CoRR, abs/1611.01578, 2016.

[Zo17] Zoph, Barret; Vasudevan, Vijay; Shlens, Jonathon; Le, Quoc V.: Learning Transferable Architectures for Scalable Image Recognition. CoRR, abs/1707.07012, 2017.