Automatic Hyperparameter Optimization for Transfer Learning on Medical Image Datasets Using Bayesian Optimization

Rune Johan Borgli, Håkon Kvale Stensland
Simula Research Laboratory, Norway
University of Oslo, Norway
{rune, haakonks}@simula.no

Michael Alexander Riegler, Pål Halvorsen
Simula Metropolitan Center for Digital Engineering, Norway
University of Oslo, Norway
{michael, paalh}@simula.no

***Abstract*—In the medical field of gastroenterology, deep learning is being explored and utilized in computer-aided diagnosis (CAD) systems. These systems assist physicians in the diagnosis of diseases and anomalies using visual data from endoscopic examinations. Deep learning has proven effective in the field [15]–[21], [23]. However, hyperparameter optimization is usually performed manually, taking a long time and with a chance of not finding the best parameters for classification accuracy. Using transfer learning, we aim for high accuracy in anomaly detection, and this paper describes a system for automatic hyperparameter optimization of convolutional neural network (CNN) models in Keras [4]. The presented system utilizes Bayesian optimization and is used to present experiments with three optimization strategies automatically optimizing hyperparameters for CNN models on two gastrointestinal datasets. Between the strategies, one was successful in achieving a high validation accuracy, while the others were considered failures. Compared to the best approaches in related work, our best models were around 10% better. With these experiments, we demonstrated that automatic hyperparameter optimization is an effective strategy for increasing performance in transfer learning.**

***Index Terms*—hyperparameter optimization, bayesian optimization, keras, medical images, gastrointestinal tract, gpyopt**

I. Introduction

In the field of medicine, technological advances may potentially improve anomaly detection rates, and in recent years, extensive research has been performed in the field of computer-aided diagnosis systems (CADs) [3], [12]. These are systems aim to aid physicians during and after examinations in diagnosing patients. CADs usually use visual, textual or sensory data to evaluate and categorize diseases. One field where the use of CADs are explored is gastroenterology [13], [18], [26]. For diseases affecting the gastrointestinal (GI) tract, colonoscopy and gastroscopy are the de facto examinations of the colon and esophagus, respectively. In both, a tube with a camera attached is inserted into body cavities, and a video stream allows the physician to examine and diagnose the patient. Here, a CAD can apply machine learning on the video streams for automatic detection and classification of anomalies, diseases, and medical procedures. However, to train these deep learning models, large amounts of annotated data should be available. Large datasets of GI images are not available as annotation requires both physicians' time and consent from the patients. To tackle this challenge, we use transfer learning.

Transfer learning is a technique for training deep learning models, where the goal is to transfer relevant knowledge from one domain to another domain [14], [29]. The target domain should usually be somehow similar to the domain transferred from. In the case of image data, convolutional neural networks (CNNs) are pre-trained on a large dataset with images which has commonalities with the lesser primary dataset. In our case, we have CNN models, pre-trained on the 2015 ImageNet Large Scale Visual Recognition Challenge dataset [7], [22] containing 1,000 classes and 150,000 images, which is fine-tuned to datasets containing images from the GI tract. The images in ImageNet are very different from the images in the GI tract datasets, but the technique is effective because all of the images are of real objects. Real objects mean lower-level image features such as colors, lines, and lighting are shared among the datasets. The benefits of transfer learning are effective training with fewer data and faster convergence during training.

CNN models consist of different convolutional layers which are applied sequentially to the input image. The convolution filter of each layer is pixel values which are tweaked while training. These weights are referred to as the model's parameters, and hyperparameters are parameters affecting the training or behavior of the model. There are different approaches to optimize the hyperparameter configuration automatically. One such solution is the Bayesian optimization, which, for example, Google has implemented in their cloud engine [5], [9]. Bayesian optimization is a sequential design strategy for global optimization of black-box functions [24]. Other solutions are grid search, random search, and Hyperband [11].

In this work, we propose a system for creating the best hyperparameter configuration for a transfer learned model in

Keras [4]. We use default Bayesian optimization implemented by the Python library GPyOpt [1]. We run two experiments, one on each of the two datasets. The system optimizes four hyperparameters, where one is unique to transfer learning. We use three different strategies with different dimensionalities of hyperparameters. Our main contributions are that (i) we present results from the experiments showing the performance of automatic hyperparameter optimization of transfer learning and (ii) we proposed and developed a system for running automatic experiments with Bayesian optimization on hyperparameters with a transfer learning approach.

The rest of the paper is organized as follows: Section II briefly presents related work on CADs and automatic hyperparameter optimization systems. Section III describes the proposed system implementation, which we use to run experiments in section IV. In section IV, we also present and discuss the results from the experiments. Section V concludes and summarizes the paper as well as presenting future work.

II. Related Work

The area of automated anomaly detection in images and videos from the GI tract, especially polyps, has grown significantly in the last years. Starting with image analysis and object detection [20], [28] (multiple approaches listed in [18]), the approaches have become more sophisticated where a number of proposals use CNNs. For example, Deep-EIR [18] is based on pre-trained CNN architectures and transfer learning. Tajbakhsh et al. [25], [26] have used transfer learning and CNN models for polyp detection and Mohammed et al. [13] have created a CNN model for polyp detection specifically.

The reported detection results are often very good with results about 90% detection accuracy [2], [13], [18], [26], [28]. However, the experiments are often performed on limited datasets using transfer learning, and often, hyperparameters are manually tuned which opens for the potential for even better results with better hyperparameter settings. An automatic hyperparameter approach is desirable as it has the potential of reaching better results without human intervention.

CNNs are to a growing degree being used for anomaly detection, but configuring the networks is a challenge. Concerning automatic optimization, there also exist some work. Auto-Weka [27], Auto-Keras [8], and Google Vizier [6] are all tools promising automatic hyperparameter optimization. However, they do not consider the transfer learning hyperparameter we use in this work, and neither the pre-trained model. There exist several techniques for optimization, but the mentioned tools all report Bayesian optimization to be the best approach which was also the reason why we chose it for our approach.

In summary, CAD systems targeting GI tract analysis are exploring transfer learning with CNN models for anomaly detection, but they are not using automatic hyperparameter optimization. Existing tools for automatic hyperparameter optimization are not built for the implementation of transfer learning we use or all the hyperparameters we use. However, we draw inspiration from these tools by using Bayesian optimization, which has become the standard for hyperparameter optimization.

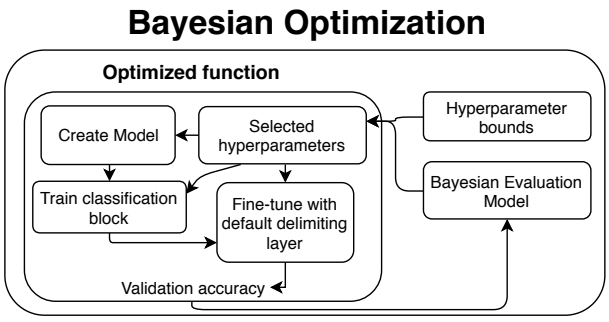


Fig. 1: Bayesian optimization in our system.

III. System Implementation

As discussed previously, optimizing CNNs is error-prone and time-consuming. Moreover, if not performed in a good way, it may not give the best detection accuracy. Automatic optimizers exist, but there are still issues in our medical context. Thus, we here present our approach for automatic hyperparameter optimization.

A. Bayesian Optimization

For our system, we use Bayesian optimization implemented in the Python library GPyOpt [1]. GPyOpt allows different parameters affecting the performance of the optimization. We use the default parameters provided by the library since we are mainly interested in researching the possible performance gain combining it with transfer learning. GPyOpt allows us to pass a function for optimization, specify how many iterations we want to optimize for, and print the acquisition graph at the end of the optimization. We also provide a list of hyperparameters and their boundaries. Increasing the number of hyperparameters also increases the dimensionality of the search space, which makes it more difficult to find good hyperparameters without also increasing the number of iterations.

The Bayesian optimization uses a surrogate model which is fitted to observations of the real model. An observation in our case is a full training of a CNN model using hyperparameters selected for that observation. For each iteration, a set of hyperparameters is chosen, and an observation made. We use validation accuracy for evaluation of the observation. The hyperparameter set is chosen based on an acquisition function balancing the choice between exploring the whole search space and exploiting well-performing areas of the search space. Figure 1 shows the flow of the optimization.

B. Transfer learning

We use transfer learning for efficient training of a relatively small dataset, and for the speed-up the technique presents over training from scratch. Our implementation of transfer learning relies on pre-trained models available from Keras [4], which have been pre-trained on the ImageNet dataset [7]. ImageNet is a dataset containing 1,000 classes of ordinary objects such as a car, house, balloon, and strawberry. These images are different from images from the gastrointestinal domain, but share the commonality of being natural images with colors, contrasts, and light. These shared image features are the ones we want

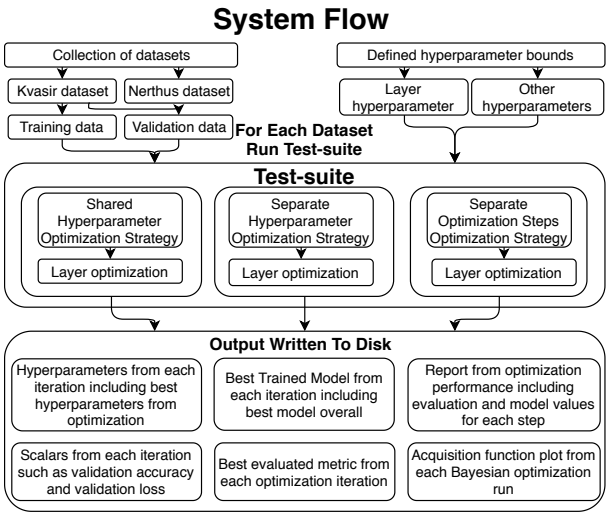


Fig. 2: System flow.

to keep while fine tuning to our gastrointestinal dataset. We have implemented this by selecting a layer in the model as a separator. All layers before this layer, those containing lower-level image features, are not changed during training, that is, frozen. We train all layers after this layer as usual which introduces a new hyperparameter that we call the *delimiting layer*.

To use transfer learning in Keras, we must replace the classification block of the CNN model. The old block was for the 1,000 classes of ImageNet, so we replaced it with a block containing our number of classes. The first step of transfer learning is then to train the new classification block alone until it is on par with the rest of the model. The last step is to fine tune the model based on the chosen delimiting layer.

C. Methodology

Figure 2 shows the flow of the system. We split the datasets into 70% training data and 30% validation data. Moreover, a set of hyperparameters and their boundaries is defined, including the delimiting layer. Further, we run the optimization test-suite. Here, we have three approaches to how we use the hyperparameters in the optimization.

1) The first approach, *shared hyperparameters optimization*, is to have one set of hyperparameters which we use for both the classification block step and the fine-tuning step of the training.
2) The second approach, *separate hyperparameters optimization*, is to use two sets of hyperparameters, one for each training step. The drawback to this approach is that we double the search space dimensionality.
3) The third approach, *separate optimization steps optimization*, is to use two sets of hyperparameters, but instead of choosing them in the same observation, we run two separate Bayesian optimization runs. First, we optimize the classification block, and then the best model is used for separate optimization of the fine-tuning.

After every optimization approach, we perform a separate optimization including only the delimiting layer. The reason for this is that we need to define the boundaries of the hyperparameter. The boundaries of the delimiting layer are from 0 to the maximum number of layers of the model used. We use the model as a hyperparameter, which means the model changes during optimization. The separate layer optimization step allows us to find the best model first, then learning the hyperparameter bounds of the delimiting layer. However, the dependency between the delimiting layer and the pre-trained model can be problematic. Before conducting layer optimization, we need to use a temporary value for the delimiting layer. Some models might work better with this default setting than others. We use their performance with the default layer to evaluate their efficiency, so we risk models to be chosen based on their performance with the default delimiting layer before layer optimization. In the future, we plan not to separate the Bayesian optimization steps to avoid dependencies like this.

IV. Experiments

To evaluate our system, we ran two experiments on two datasets testing each of the three previously mentioned optimization strategies. Each Bayesian optimization step ran with ten iterations as the maximum. Additionally, to save time, some training runs were canceled earlier because they failed to converge beyond 50% validation accuracy after the completion of five epochs. Our hardware specifications include two Intel Xeon E5-2630 v3 2.40GHz CPUs, one Nvidia Tesla K40c and two Nvidia GeForce GTX TITAN X GPUs, and 64 GB system memory.

For the experiments presented here, we chose to focus on four hyperparameters: The pre-trained model, the gradient descent optimizing function, the learning rate and the delimiting layer. The pre-trained model is a hyperparameter because Keras keeps a number of models, which can be swapped seamlessly. For a user of machine learning, choosing the best model will often be based on benchmark numbers, but these are not necessarily representative for the given dataset. We use the following pre-trained models available in Keras: Xception, VGG16, VGG19, ResNet50, InceptionV3, InceptionResNetV2, DenseNet121, DenseNet169, and DenseNet201. For the gradient descent optimizer, we use the following available in Keras: SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, and Nadam. We set the learning rate bounds as continuous between 1 and 10^{-4}. Lastly, as previously mentioned, the delimiting layer is set between 0 and the number of layers in the chosen model. Before performing the layer optimization, we use a default delimiting layer of $2/3$ of the number of layers in the model.

A. Datasets

The Kvasir dataset [17] consists of eight classes as shown in Figure 3. The dataset is balanced with each class having 1,000 images. Some images in certain classes contain artifacts such as a box or text. These artifacts could affect the generalization capability of the trained models. For our purposes, the effect

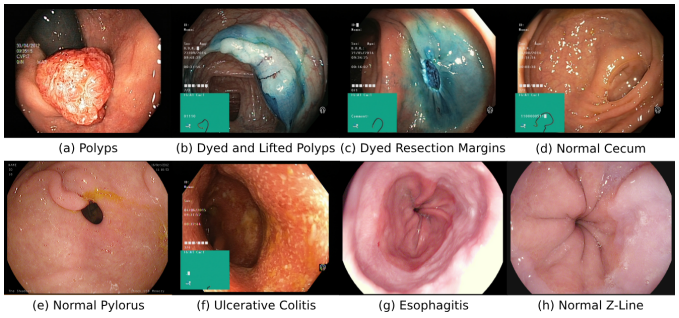


Fig. 3: Example images from each class in Kvasir [17].

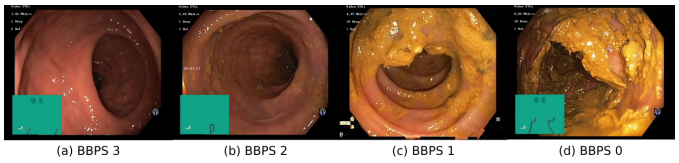


Fig. 4: Example images from each class in Nerthus [16].

is irrelevant as we are only interested in increased detection compared to the baseline approaches presented in the paper describing the dataset.

The Nerthus dataset [16] is different from Kvasir in both that we have videos instead of separate images and the four classes, shown in Figure 4, contain images with different values of the Boston Bowel Preparation Scale (BBPS) [10]. Nerthus contains the same artifacts as Kvasir. Nerthus consists of 21 videos with a total number of 5,525 frames and is an imbalanced dataset with frames per class varying from 500 to 2,700.

B. Evaluation Metrics

We use the metrics suggested in the dataset papers [16], [17]: F1 score (F_1), accuracy (ACC), Matthew correlation coefficient (MCC), precision (PREC), recall/sensitivity (REC), and specificity (SPEC). A detailed description and reasoning for the use of these metrics for evaluation can be found in [17]. Additionally, we report false negatives and positives, and true negatives and positives. Lastly, we use validation accuracy to determine the performance of the model during training.

C. Results and Discussion

1) Kvasir: Figure 5 shows the experiment ran for Kvasir. In the graph, each line represents a training step's validation accuracy over time. The experiment took almost four days. We can see in Figure 5 that the shared hyperparameters optimization strategy outperformed the other two optimization strategies regarding validation accuracy. Shared hyperparameters achieved a validation accuracy of 0.89, which was much higher than the best of the separate hyperparameters at 0.63 and the best of the separate optimizations at 0.54 validation accuracy. The classification block optimization reached a validation accuracy of 0.86 and the layer optimization increased the validation accuracy to 0.89, which suggests that optimizing the delimiting layer makes a difference.

Additionally, we see that the shared hyperparameters optimization is the only strategy to converge to a higher validation

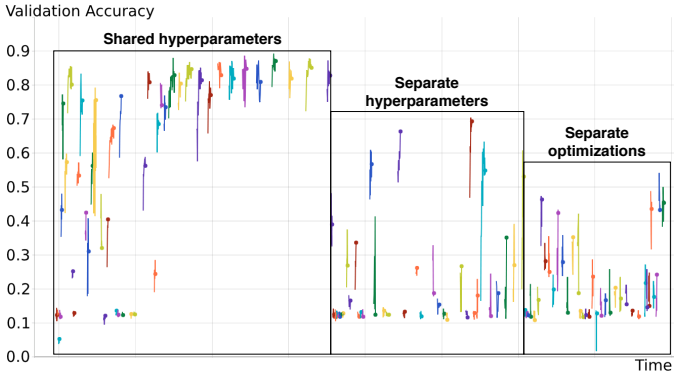


Fig. 5: The experiment on the Kvasir dataset.

Class	FN	FP	TN	TP	F_1	ACC	MCC	PREC	REC	SPEC
Dyed lifted polyps	15	24	2076	285	0.94	0.98	0.93	0.92	0.95	0.99
Dyed resection margins	23	8	2092	277	0.95	0.99	0.94	0.97	0.92	1.00
Esophagitis	181	4	2096	119	0.56	0.92	0.59	0.97	0.40	1.00
Normal cecum	8	18	2082	292	0.96	0.99	0.95	0.94	0.97	0.99
Normal pylorus	2	18	2082	298	0.97	0.99	0.96	0.94	0.99	0.99
Normal z-line	7	183	1917	293	0.76	0.92	0.74	0.62	0.98	0.91
Polyps	19	7	2093	281	0.96	0.99	0.95	0.98	0.94	1.00
Ulcerative colitis	20	13	2087	280	0.94	0.99	0.94	0.96	0.93	0.99
Average	34	34	2066	266	0.88	0.97	0.87	0.91	0.89	0.98
Best from [17]	-	-	-	-	0.75	0.94	0.71	0.75	0.75	0.96

TABLE I: Metrics for each class after hyperparameter optimization for the best model on the Kvasir dataset, including average values for comparing with the best approach metrics presented in the Kvasir paper [17].

accuracy. The other strategies are more sporadic. It seems luck has been the big factor in their best results. We can also observe that many of the runs failed to reach above 0.5, with most runs being around 0.125, which means random classification. All of these runs were stopped early, and we can see the impact of this saving us time.

In Table I, we compare the best-trained model from the Kvasir dataset experiment with the best approach from the Kvasir dataset paper [17]. The best-trained model was trained from the shared hyperparameters optimization strategy in the layer optimization step and reached a validation accuracy of 0.89. The best-trained model reached an F_1 score of 0.88 which was significantly better than the best from Kvasir with a score of 0.75, which was a method using handcrafted global image features. Compared to a transfer learning method using InceptionV3 achieving an F_1 score of 0.69, which did not include hyperparameter optimization, our results show that hyperparameter optimization makes a significant impact on the accuracy of the classifier.

Our best-trained model did significantly better than the comparable approaches in the Kvasir dataset paper. By looking at the confusion matrix in Figure 6, we see that the model classifies most images correctly. However, we can see between two pairs of classes that the model makes misclassifications. Between dyed lifted polyps and dyed resection margins, there are a few misclassifications, but we attribute this to both

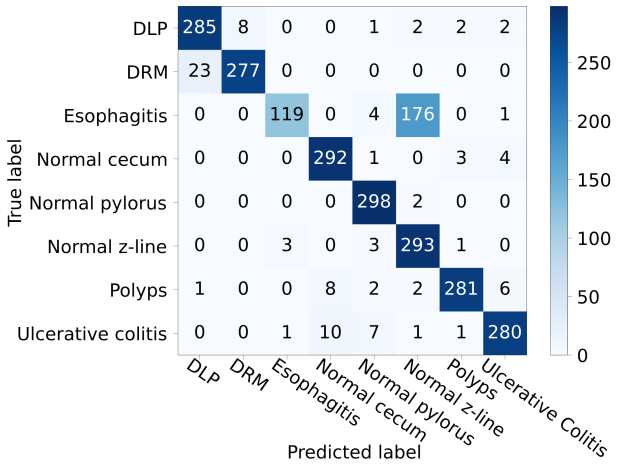
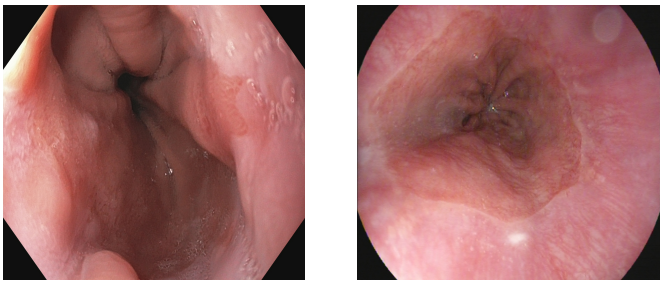


Fig. 6: Confusion matrix: best-trained model on Kvasir.



(a) Esophagitis misclassified as normal-z-line.

(b) Normal-z-line misclassified as esophagitis.

Fig. 7: Misclassification: esophagitis and normal-z-line.

classes having blue dye covering the area of interest. Also, a few of the images contain both dyed lifted polyps and dyed resection margins which confuse the classifier. Between esophagitis and normal-z-line, the esophagitis class is wrongly classified in 176 instances. This misclassification is the only reason why the average F_1 score is not above 0.95. However, by looking at Figure 7, we can see examples of these misclassifications. We see from the figure that there are visual similarities which might even make it difficult for humans to distinguish. It is our opinion that the similarities are so striking, that the algorithm, similar to a human, is unable to spot the differences. We have one question: If some images are so similar that they are interchangeable, why are there almost no misclassifications where the normal z-line class is classified as the esophagitis class? By looking at the confusion matrix in Figure 6, we see the misclassifications where the normal z-line class is classified as the esophagitis class involving only three images. We can only speculate that it must be of some CNN implementation-specific reason.

2) Nerthus: From Figure 8, we observe many of the same patterns as in Figure 5 from the Kvasir experiments. The shared hyperparameters optimization approach is the winner regarding validation accuracy. However, we see that the separate optimization strategy performed close to the winner. What stands out in the Nerthus runs is that the validation accuracy of the best-trained model reached very close to 100%. Layer optimization nudged the results from 0.99 to 1. The high result might be due to overfitting on the validation dataset,

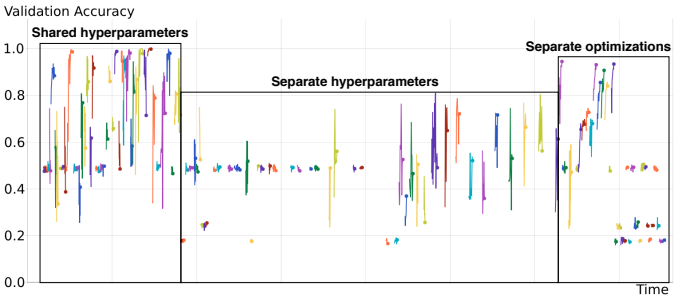


Fig. 8: The experiment on the Nerthus dataset.

Class	FN	FP	TN	TP	F_1	ACC	MCC	PREC	REC	SPEC
BBPS 0	0	0	1507	150	1.00	1.00	1.00	1.00	1.00	1.00
BBPS 1	1	2	845	809	1.00	1.00	1.00	1.00	1.00	1.00
BBPS 2	0	1	1364	292	1.00	1.00	1.00	1.00	1.00	1.00
BBPS 3	2	0	1252	403	1.00	1.00	1.00	1.00	1.00	1.00
Average	0.75	0.75	1242	414	1.00	1.00	1.00	1.00	1.00	1.00
Best from [16]	-	-	-	-	0.90	0.95	0.86	0.90	0.90	0.96

TABLE II: Metrics for each class after hyperparameter optimization for the best model on the Nerthus dataset, including average values for comparing with the best approach metrics presented in the Nerthus paper [16].

but avoiding overfitting is not a goal of the experiments. Another reason for high results might be a high correlation between frames of the same video. We treat each frame as independent to make our results comparable to the approaches in the Nerthus dataset paper [16].

In Figure 8, we see that the separate hyperparameters optimization took longer than the others, but did worse. We see this is in contrast to what we saw in Figure 5 for the Kvasir experiments. The reason for this is that most training runs were centered around 0.5 validation accuracy, which meant they were not stopped early. For this experiment, we should have increased the threshold for being stopped early. Still, the whole run took almost two days, which makes sense as the dataset is smaller than Kvasir. We also see that the separate hyperparameters and optimizations strategies failed again at optimizing as none converged to as high validation accuracy as the shared hyperparameters. The failure is likely due to the higher dimensionality of the two strategies compared to sharing the hyperparameters.

We had several models reaching near 100% validation accuracy. Therefore, we had to choose one of them. In Table II, we compare the evaluation metrics to those for the best approach in the Nerthus dataset paper [16]. An almost perfect classification is tough to beat, and, indeed, we see in Table II, that the metrics from our best model are much better than the metrics produced in the Nerthus paper [16]. The overall metrics are higher than those in the Kvasir paper [17], shown in Table I, meaning classification is overall easier for Nerthus than for Kvasir.

Reaching better metrics for the best Nerthus model is, therefore, expected. The best F_1 score in the Nerthus paper was reached by the six global features logistic model tree approach, the same method that reached the best result in the

Kvasir paper, reaching 0.9. In comparison to our model, that is 10% less. The more similar transfer learning with InceptionV3 approach in the Nerthus paper reached an F_1 score of 0.75, which is significantly worse. We can see from our results that our shared hyperparameters optimization approach can produce better significantly better classifiers than approaches using handcrafted global features or transfer learning without automatic hyperparameter optimization.

V. Conclusion

For the scenario of anomaly detection in the GI tract, we have presented a system for automatic hyperparameter optimization using transfer learning in Keras. We ran experiments using two datasets containing images and videos from the GI tract, and we evaluated three different hyperparameter optimization strategies. The results indicates that only the shared hyperparameters approach was successful. For the shared hyperparameters approach, the results for both experiments showed an increase of about 10% over the best approaches in the dataset papers which we used for comparison. For similar transfer learning approaches in the dataset papers, the difference was even greater. We therefore conclude that automatic hyperparameter optimization is an effective strategy for increasing performance in transfer learning use cases. For future work, we suggest to remove the pre-trained model as a hyperparameter as we believe that we could achieve better performance by having the pre-trained model and the delimiting layer in the same optimization. Another important future work is to experiment the effect of overfitting in the approaches presented and try different configurations of Bayesian optimization.

References

[1] The GPyOpt authors. Gpyopt: A bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt, 2016.

[2] Rune Johan Borgli, Pål Halvorsen, Michael Riegler, and Håkon Kvale Stensland. Automatic hyperparameter optimization in keras for the mediaeval 2018 medico multimedia task. In *Working Notes Proc. of the MediaEval 2018 Workshop*. CEUR Workshop Proceedings (CEUR-WS.org), 2018.

[3] Michael F Byrne, Neal Shahidi, and Douglas K Rex. Will Computer-Aided Detection and Diagnosis Revolutionize Colonoscopy? *Gastroenterology*, 153(6):1460–1464.e1, dec 2017.

[4] François Chollet and et al. Keras, 2015. https://keras.io.

[5] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google Vizier: A Service for Black-Box Optimization. *Knowledge Discovery and Data Mining (KDD)*, 2017.

[6] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Elliot Karro, and D. Sculley, editors. *Google Vizier: A Service for Black-Box Optimization*, 2017.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. of IEEE CVPR*, pages 248–255, 2009.

[8] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: Efficient neural architecture search with network morphism. *CoRR*, 2018.

[9] Greg Kochanski, Daniel Golovin, John Karro, Benjamin Solnik, Subhodeep Moitra, and D Sculley. Bayesian Optimization for a Better Dessert. In *Proc. of NIPS Workshop on Bayesian Optimization*, 2017.

[10] Edwin J Lai, Audrey H Calderwood, Gheorghe Doros, Oren K Fix, and Brian C Jacobson. The Boston bowel preparation scale: a valid and reliable instrument for colonoscopy-oriented research. *Gastrointestinal endoscopy*, 69(3 Pt 2):620–5, mar 2009.

[11] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 2016.

[12] Masashi Misawa, Shin-ei Kudo, Yuichi Mori, Tomonari Cho, Shinichi Kataoka, Akihiro Yamauchi, Yushi Ogawa, Yasuharu Maeda, Kenichi Takeda, Katsuro Ichimasa, Hiroki Nakamura, Yusuke Yagawa, Naoya Toyoshima, Noriyuki Ogata, Toyoki Kudo, Tomokazu Hisayuki, Takemasa Hayashi, Kunihiko Wakamura, Toshiyuki Baba, Fumio Ishida, Hayato Itoh, Holger Roth, Masahiro Oda, and Kensaku Mori. Artificial Intelligence-Assisted Polyp Detection for Colonoscopy: Initial Experience. *Gastroenterology*, 154(8):2027–2029.e3, jun 2018.

[13] Ahmed Kedir Mohammed, Sule Yildirim, Ivar Farup, Marius Pedersen, and Øistein Hovde. Y-net: A deep convolutional neural network for polyp detection. *CoRR*, abs/1806.01907, 2018.

[14] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning, oct 2010.

[15] Konstantin Pogorelov, Sigrun Losada, Carsten Griwodz, Thomas de Lange, Kristin Ranheim Randel, Duc Tien Dang Nguyen, Håkon Kvale Stensland, Francesco De Natale, Dag Johansen, Michael Riegler, and Pål Halvorsen. A Holistic Multimedia System for Gastrointestinal Tract Disease Detection. In *Proc. of ACM MMSys*, 2017.

[16] Konstantin Pogorelov, Kristin Ranheim Randel, Thomas de Lange, Sigrun Losada Eskeland, Carsten Griwodz, Dag Johansen, Concetto Spampinato, Mario Taschwer, Mathias Lux, Peter Thelin Schmidt, Michael Riegler, and Pål Halvorsen. Nerthus: A bowel preparation quality video dataset. In *Proc. of ACM MMSys*, pages 170–174, 2017.

[17] Konstantin Pogorelov, Kristin Ranheim Randel, Carsten Griwodz, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Concetto Spampinato, Duc-Tien Dang-Nguyen, Mathias Lux, Peter Thelin Schmidt, Michael Riegler, and Pål Halvorsen. Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection. In *Proc. of ACM MMSys*, pages 164–169, 2017.

[18] Konstantin Pogorelov, Michael Riegler, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Carsten Griwodz, Peter Thelin Schmidt, and Pål Halvorsen. Efficient disease detection in gastrointestinal videos global features versus neural networks. *Multimedia Tools and Applications*, 76(21):22493–22525, 2017.

[19] Konstantin Pogorelov, Michael Riegler, Pal Halvorsen, Peter Thelin Schmidt, Carsten Griwodz, Dag Johansen, Sigrun Losada Eskeland, and Thomas de Lange. GPU-Accelerated Real-Time Gastrointestinal Diseases Detection. In *Proc. of IEEE CBMS*, pages 185–190, jun 2016.

[20] Michael Riegler, Mathias Lux, Carsten Griwodz, Concetto Spampinato, Thomas de Lange, Sigrun L Eskeland, Konstantin Pogorelov, Wallapak Tavanapong, Peter T Schmidt, Cathal Gurrin, Dag Johansen, Håvard Johansen, and Pål Halvorsen. Multimedia and Medicine: Teammates for Better Disease Detection and Survival. In *Proc. of ACM MM*, pages 968–977, 2016.

[21] Michael Alexander Riegler. *EIR - A Medical Multimedia System for Efficient Computer Aided Diagnosis*. PhD thesis, University of Oslo, 2017.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[23] Klaus Schoeffmann, Bernd Münzer, Michael Riegler, and Pål Halvorsen. Medical Multimedia Information Systems (MMIS). In *Proc. of ACM MM*, pages 1957–1958, 2017.

[24] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *Adv. Neural Inf. Process. Syst. 25*, pages 1–9, 2012.

[25] N. Tajbakhsh, S. R. Gurudu, and J. Liang. Automatic polyp detection in colonoscopy videos using an ensemble of convolutional neural networks. In *Proc. of IEEE ISBI*, pages 79–83, April 2015.

[26] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, May 2016.

[27] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proc. of ACM KDD*, 2013.

[28] Yi Wang, Wallapak Tavanapong, Johnny Wong, Jung Hwan Oh, and Piet C. de Groen. Polyp-Alert: Near real-time feedback during colonoscopy. *Computer Methods and Programs in Biomedicine*, 120(3):164–179, jul 2015.

[29] Karl Weiss, Taghi M. Khoshgoftaar, and Ding Ding Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, dec 2016.