Automatic Hyperparameter Optimization in Keras for the MediaEval 2018 Medico Multimedia Task

Rune Johan Borgli, Pål Halvorsen, Michael Riegler, Håkon Kvale Stensland
Simula Research Laboratory, Norway
rune@simula.no,paal@simula.no,michael@simula.no,haakonks@simula.no

ABSTRACT

This paper details the approach to the MediaEval 2018 Medico Multimedia Task made by the Rune team. The decided upon approach uses a work-in-progress hyperparameter optimization system called Saga. Saga is a system for creating the best hyperparameter finding in Keras [5], a popular machine learning framework, using Bayesian optimization and transfer learning [3]. In addition to optimizing the Keras classifier configuration, we try manipulating the dataset by adding extra images in a class lacking in images and splitting a commonly misclassified class into two classes.

1 INTRODUCTION

We made the following approach as a submission to the Mediaeval 2018 Medico Multimedia Task. The task contains several sub-tasks, but we have focused solely on the detection sub-task. Information about the given dataset, task, and evaluation are described in the Medical Multimedia Task overview paper [13]. In short, the task is to create a classifier for a 16-class dataset containing a total of 5277 images from the medical domain of the digestive system.

2 APPROACH

Our approach is a two-split approach. First, our main contribution is to use automatic hyperparameter optimization building on Borgli's thesis [3] to find a hyperparameter configuration in Keras for our classifier submission. Second, we tried a few alterations of the dataset to see if we could improve the classifier performance further.

2.1 Hyperparameter optimization

Automatic hyperparameter optimization was done using an early, unpublished, work-in-progress system called Saga. Saga is a tool with a web-interface providing developers of image-based machine learning applications an easy, customizable work-flow for creating well-performing classifiers for image data. The user only needs to supply the training and validation dataset. Everything else is provided by either a Keras [5] or a Pytorch [10] back-end. We split the system into three: (1) preprocessing of the dataset and a metric for the predicted effectiveness of the dataset, (2) running any valid configuration of hyperparameters or hyperparameter optimizations available in Keras or Pytorch for training of the classifier on a given dataset, and (3) visualization and analysis of the training and its outcome. For this submission, Keras was used together with TensorFlow [1] with no preprocessing of the dataset and no analysis of the results as we have not implemented the implementation of Pytorch, preprocessing, and analysis as of writing this paper.

MediaEval'18, 29-31 October 2018, Sophia Antipolis, France

Due to the nature of a working paper, we will only briefly describe the system. More details about how we implemented the hyperparameter optimization can be found in [3].

Our target domain is image datasets. Convolutional Neural Networks (CNNs) are types of machine learning models that excel at classifying images. However, these types of models require enough training data to avoid generalization issues when training. The provided dataset for the Medico Multimedia Task [11, 12] contains about 5000 images which are a low number compared to common benchmark datasets such as ImageNet [8]. To accommodate for this, we use transfer learning. Transfer learning is a technique where, instead of training a model from scratch, we pre-train the model using a different dataset in a similar domain, then fine-tune the model to our dataset. The idea behind transfer learning is to transfer relevant knowledge learned from the pre-training instead of learning it from scratch. Transfer learning works for different datasets because they often share basic image features, and has the added benefit that training is significantly faster. Achieving this is easy in Keras as we can use models pre-trained on ImageNet available from the framework.

We use Bayesian optimization for the automatic hyperparameter optimization. Bayesian optimization uses a surrogate function to map the function we try to optimize. Based on sequential observations, where one observation is a training run with hyperparameters chosen by the optimization, Bayesian optimization fits the surrogate function to the function to optimize. An acquisition function decides where in the search space to try the next observation based on a balance between exploration and exploitation. Exploration tries to explore the whole search space, and exploitation tries to slightly adjust the observations in those parts of the search space where results are good. Bayesian optimization can optimize any number and type of hyperparameters, but observations are costly, so we limit the dimensionality and size of the search space. We use a framework called GPyOpt for our implementation of Bayesian optimization [2] and use default parameters for the optimization function. The optimization is done after a given number of observations, and the best hyperparameters are the ones from the observation achieving the highest validation accuracy.

2.2 Dataset manipulation

Besides the hyperparameters, the performance of the classifier is dependent on the training set. Therefore, to try to improve our classifier's performance, we made two separate tweaks to the dataset. First, based on an observation that the esophagitis dataset contains images of both the upper and middle esophagus and the diseased z-line, we wanted to see if extracting the esophagitis z-line images in a separate class could increase the detection rate between

Model	Optimizer	Layer	Val Acc
DenseNet121 [7]	SGD	0	0.942
DenseNet169 [7]	SGD	0	0.954
DenseNet201 [7]	Adamax [9]	356	0.651
InceptionResNetV2 [15]	Adamax [9]	395	0.89
InceptionV3 [16]	Adadelta [18]	68	0.793
ResNet50 [6]	Adadelta [18]	2	0.911
VGG16 [14]	SGD	10	0.921
VGG19 [14]	SGD	0	0.93
Xception [4]	RMSprop [17]	0	0.916

Table 1: Results of running initial hyperparameter optimization on models available in Keras. Hyperparameters tweaked for each model were the gradient descent optimizer and the delimiting layer.

esophagitis and normal z-line. The idea behind this is that the classifier could become "confused" by seeing very different images of the same class. Secondly, we added images to the out-of-patient class to increase the performance. These images were of medical equipment in medical rooms.

3 RESULTS AND DISCUSSION

The hyperparameters we optimized were the model, the gradient descent optimizer, the learning rate and the delimiting layer. First, we ran automatic hyperparameter optimization for the gradient descent optimizer and delimiting layer for each model to have an initial map of the performance of each model. The results are listed in table 1. Furthermore, we picked the best performing hyperparameter configuration and ran a new optimization only optimizing the learning rate for the first and second step of the transfer learning. The configuration was model DenseNet169, gradient descent optimizer SGD, and delimiting layer of 0, meaning we fine tune all of the layers. The best classifier's learning rate for the block optimization ended on 0.0001, and the fine tuning ended on 0.00067, with a validation accuracy of 0.954. The second best classifier's learning rate for the block optimization ended on 0.000046, and the fine tuning ended on 0.004, with a validation accuracy of 0.952.

After hyperparameter optimization, we trained classifiers on a dataset where esophagitis images of the z-line were extracted into a separate class. We trained the classifiers using DenseNet169, SGD, a delimiting layer of 0, and a learning rate default to SGD, which is 0.01. The results for the best classifier were a validation accuracy of 0.928 and for the second best classifier a validation accuracy of 0.925. For the classifier where we added more images to the out of patient class, we ran the same classifier setup and got a validation accuracy of 0.925.

All submissions can be found with their resulting Matthews correlation coefficient (MCC) score in figure 2. For both the hyperparameter optimization and the split class, we see that the second best submission performed better. This observation indicates that the best runs overfitted on the validation set. For future experiments, measures such as splitting into a third test set or using k-fold cross-validation should be applied to the results to avoid this issue.

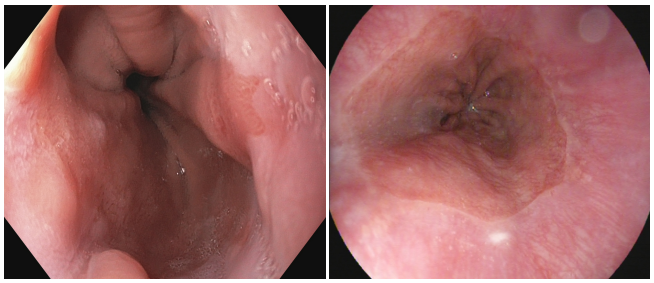
Lastly, table 3 shows the confusion matrix for the best result on the test set. We can see that many of the classes have very high accuracy, and a few pairs of classes have many misclassifications between them. We speculate that this is due to the nature of the dataset where several classes are very different while others are very similar. An example of misclassification and the similarities between classes can be observed in figure 1.

Table 2: Matthews Correlation Coefficient for each submitted approach for the validation set after training and the test set used for the competition results.

Approach	Result MCC
Best hyperparameter optimization	0.927
Second best hyperparameter optimization	0.931
Best split esophagitis class	0.916
Second best split esophagitis class	0.920
Added data to out of patient class	0.925

Table 3: Result confusion matrix for the best performing classifier. The X-axis is predicted labels, Y-axis is true labels. The labels are as follows: (A) Ulcerative colitis, (B) Esophagitis, (C) Normal z-line, (D) Dyed lifted polyps, (E) Dyed resection margins, (F) Out of patient, (G) Normal pylorus, (H) Stool inclusions, (I) Stool plenty, (J) Blurry nothing, (K) Polyps, (L) Normal cecum, (M) Colon clear, (N) Retroflex rectum, (O) Retroflex stomach, and (P) Instruments.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	521	0	0	0	2	0	2	3	70	0	4	8	1	3	0	18
B	1	434	66	0	0	0	0	0	0	0	0	0	0	0	0	0
C	1	119	496	0	0	0	1	0	0	0	0	0	0	0	1	0
D	0	0	0	506	32	0	0	0	0	0	1	0	0	0	0	36
E	0	0	0	41	528	0	0	0	0	0	1	0	0	0	0	16
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	1	546	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	493	12	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	1873	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0
K	14	3	1	8	1	0	11	0	7	0	364	14	0	2	0	73
L	4	0	0	1	0	0	0	0	0	0	4	562	0	0	0	1
M	0	0	0	0	0	0	0	8	1	0	0	0	1064	0	0	0
N	1	0	0	0	0	1	0	0	0	0	0	0	0	182	1	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	5	395	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	124



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

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

Figure 1: Examples of misclassification between esophagitis and normal-z-line.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). https://www.tensorflow.org/ Software available from tensorflow.org.

[2] The GPyOpt authors. 2016. GPyOpt: A Bayesian Optimization framework in python. http://github.com/SheffieldML/GPyOpt. (2016).

[3] Rune Johan Borgli. 2018. Hyperparameter optimization using Bayesian optimization on transfer learning for medical image classification. (2018). Master thesis at University of Oslo. https://www.duo.uio.no/handle/10852/64146.

[4] François Chollet. 2016. Xception: Deep Learning with Depthwise Separable Convolutions. *CoRR* abs/1610.02357 (2016). arXiv:1610.02357 http://arxiv.org/abs/1610.02357

[5] François Chollet and others. 2015. Keras. https://keras.io. (2015).

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[7] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPRW.2009.5206848

[9] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Conference on Neural Information Processing Systems*.

[11] 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. 2017. Nerthus: A Bowel Preparation Quality Video Dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 170–174.

[12] 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. 2017. Kvasir: A Multi-Class Image Dataset for Computer Aided Gastrointestinal Disease Detection. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 164–169.

[13] Konstantin Pogorelov, Michael Riegler, Pål Halvorsen, Thomas de Lange, Kristin Ranheim Randel, Duc-Tien Dang-Nguyen, Mathias Lux, and Olga Ostroukhova. 2018. Medico Multimedia Task at MediaEval 2018. In *CEUR Proceeding of the MediaEval Benchmark*.

[14] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014). arXiv:1409.1556 http://arxiv.org/abs/1409.1556

[15] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. 2016. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *CoRR* abs/1602.07261 (2016). arXiv:1602.07261 http://arxiv.org/abs/1602.07261

[16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the Inception Architecture for Computer Vision. *CoRR* abs/1512.00567 (2015). arXiv:1512.00567 http://arxiv.org/abs/1512.00567

[17] T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning. (2012).

[18] Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR* abs/1212.5701 (2012). arXiv:1212.5701 http://arxiv.org/abs/1212.5701