FOCAL LOSS FOR ARTEFACT DETECTION IN MEDICAL ENDOSCOPY

Maxime Kayser[1], *Roger D. Soberanis-Mukul*[1], *Shadi Albarqouni*[1], *Nassir Navab*[1]

[1]Chair for Computer Aided Medical Procedure (CAMP), Technische Universität Mnchen (TUM), Munich, Germany

ABSTRACT

Endoscopic video frames tend to be corrupted by various artefacts impairing their visibility. Automated detection of these artefacts will foster advances in computer-assisted diagnosis, post-examination procedures and frame restoration software. In this work, we propose an ensemble of deep learning object detectors to automate multi-class artefact detection in video endoscopy. Our approach achieved a mean average precision (mAP) of 0.3087 and an average intersection-over-union (IoU) of 0.3997 on the EAD2019 test set. This resulted in a final score of 0.3451 and the 3rd rank in the EAD 2019 object detection sub-challenge leaderboard.

***Index Terms*—** Endoscopy, RetinaNet, artefact detection, deep learning, object detection

1. INTRODUCTION

Tissue characteristics of hollow organs, as well as the different instruments and illumination modes applied in medical endoscopy lead to a high number of artefacts that obstruct visibility. The frames produced during endoscopic interventions can be corrupted by bubbles, instruments and image deficiencies such as specular reflections, strong contrasts, saturated pixels, motion blurs, or other artefacts. These corruptions have a negative effect on both the live diagnosis and post-intervention procedures. Successfully detecting the artefacts will thus assist endoscopic experts and will be the cornerstone of successful endoscopic video frame restoration.

Over recent years deep learning techniques have become the state-of-the-art in medical image analysis and they have in particular proved successful in related medical endoscopy computer vision tasks, such as polyp detection [1]. Deep learning object detection methods can be separated into one- and two-stage methods. One-stage methods are generally faster as they dont rely an additional region proposal step. RetinaNet is a one-stage detector proposed by Lin et al. [2]. The novelty of this architecture is the proposed focal loss, which addresses the imbalance between foreground and background anchors that occurs in one-stage methods. RetinaNet outperforms two-stage methods such as Faster R-CNN on COCO test-dev. We use this network as the base architecture for our challenge submission. The EAD dataset [3, 4] is very unbalanced and contains objects at vastly different scales. RetinaNets built in Feature Pyramid Network (FPN) [5] and focal loss can effectively address these issues.

2. METHODS

Our method consists of an ensemble of seven RetinaNet architectures that vary in hyperparameters, backbone networks, transfer learning, data augmentation, and training subset used. The models are combined based on an efficient voting scheme.

2.1. RetinaNet architecture

The RetinaNet detector consists of a backbone network for extracting a convolutional feature map and two subnetworks that perform object classification and bounding box regression via convolution. The classification loss is given by the focal loss and the regression loss is given by the smooth L1 loss. The sum of both these losses constitutes the overall loss that is minimized during training. It is a one-stage method, meaning that it does not require a region proposal module. Instead, anchors at different scales and aspect ratios are densely distributed across the image and they will all be classified by the network. In order to construct a multi-scale feature pyramid from a single resolution input image, the backbone network is augmented by a feature pyramid network, FPN [5]. FPNs are a top-down architecture with lateral connections that allows semantically rich layers to be built at all scales with marginal computational cost. FPNs have proven especially effective in the detection of small objects and are therefore well-suited for our use case. Pyramid levels and anchors were generated according to the specifications in [2]. We experimented with different IoU thresholds for assigning an anchor to a ground-truth object and validated the thresholds used in [2]. No other changes were made to the RetinaNet classification and regression subnetworks.

In our experiments, we used both VGGNet [6] and ResNet [7] convolutional neural networks (CNN) as the backbone network in our framework. VGGNet is a CNN from 2014 with a simple architecture that consists of convolution layers, pooling layers, and fully connected layers. We tested both a 16 and 19-layer VGGNet (Table. 2). ResNets

are much deeper CNNs that maintain their generalization capability through inception modules. Given that ResNet are much deeper and can extract more elaborate features, it generally outperforms VGGNet on most public validation test sets. We experimented with 50, 101, and 152-layered versions of ResNet.

2.2. Focal Loss

Focal Loss is an extension of the cross-entropy loss that uses a weighting factor to prevent one-stage detection methods being overwhelmed by the large amount of 'easy' background examples. Typically, one-stage methods have around 100k anchors per image. Most of these are background anchors that are easy to classify and swamp the classifier, undermining its ability to focus and learn on the harder, foreground examples. This imbalance is countered by the addition of a weighting factor $(1 - p_t)$, which reduces the weight of easily classified anchors and thereby shifts the focus onto harder examples. p_t is given in [2] as:

$$p_t = \begin{cases} p, & \text{if } y = 1 \\ p - 1, & \text{otherwise} \end{cases} \tag{1}$$

where γ is a tunable hyperparameter that modifies the extent to which the loss function prioritizes hard examples. If $\gamma = 0$, then our loss function is equal to the cross-entropy loss and no priority is given to hard examples. If for instance $\gamma = 2$ and $p_t = 0.9$ for a given anchor, then its contribution to the loss will be 100 times lower than for the standard cross-entropy loss. Our experiments (Table. 3) show that setting $\gamma = 1.5$ yields the best performance.

Besides the focal loss weighting factor, a further weighting factor α was applied. Correctly classified anchors are weighted by α and misclassified ones are weighted by $1 - \alpha$ with $\alpha \in [0, 1]$. α_t is defined analogously to p_t. According to [2], α needs to be selected together with γ. Accordingly, we set α to 0.25 for $\gamma = 1.5$.

The final α-balanced version of the focal loss is given by:

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{2}$$

2.3. Ensemble Method

In order to counter the variance in the network output and increase performance, we implemented an ensemble method. Let M be the number of models used in the ensemble method. Our final method used $M = 7$ models.

The single trained models were first ordered according to their individual test scores (Table. 5). We then iterated through the $M - 1$ first models and compared for each model m_i ($i \leq M - 1$) its bounding box predictions to all subsequent models m_j ($j > i$) in a pairwise manner. For example with three models, m_1 would be compared to m_2 and m_3. In the next iteration step m_2 would be compared to m_3 (Fig. 1).

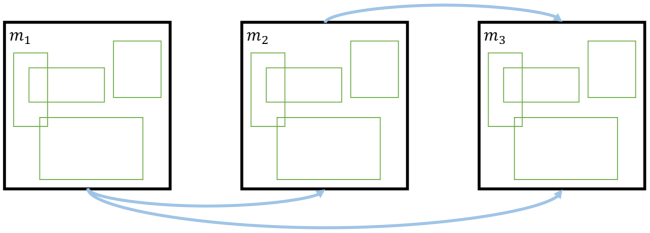

Fig. 1: Illustration of how bounding box predictions predicted by different models m_i are compared to each other. Models are first ordered in descending order of test performance. Then each model m_i is compared to all subsequent models, m_j for $j > i$.

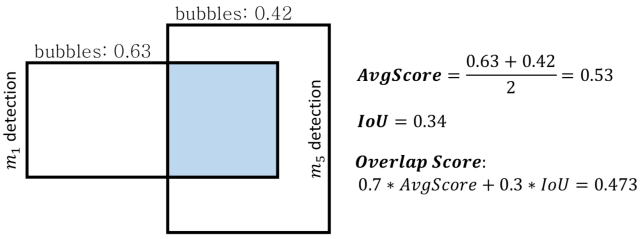

Fig. 2: Illustration of the *overlap score* computation. Displayed are two bounding box predictions from a model m_1 and a model m_5 that both predict the class bubbles. Their confidence scores are averaged. The resulting average score together with the IoU gives an overlap score of 0.473. In our case, where the threshold for determing a positive overlap is set as 0.46, this means that both boxes 'overlap' and will be assigned to each other.

Each prediction box from m_i forms the *root* of a stack and boxes from m_j can then be assigned to that stack. Whenever a bounding box from m_j is assigned to the stack of a box in m_i, it will be removed entirely to avoid assigning one box to multiple stacks. Having the most accurate bounding boxes as *roots* of the stack proved beneficial. Therefore it is important to first order the models according to their test score to achieve optimal results. Boxes are assigned to a stack based on the weighted sum of their combined average objectness confidence score and IoU with the *root* of that stack. Our *overlap score* combines the average confidence score and the IoU. Our experiments have shown that the *overlap score* computed from weighting the average score by 0.7 and IoU by 0.3 respectively yielded the best performance. Each time we compare a model m_i to a model m_j, we evaluated the *overlap score* between all of their bounding boxes and assigned boxes to each other in descending order of *overlap score*. For instance, if bounding box A from m_i and bounding box B from m_j have an overlap score of 0.92 and that is the highest score between all boxes from these two models, then B is assigned to the stack of A and B is no longer considered in future comparisons. We used a threshold overlap score of 0.46 to assign

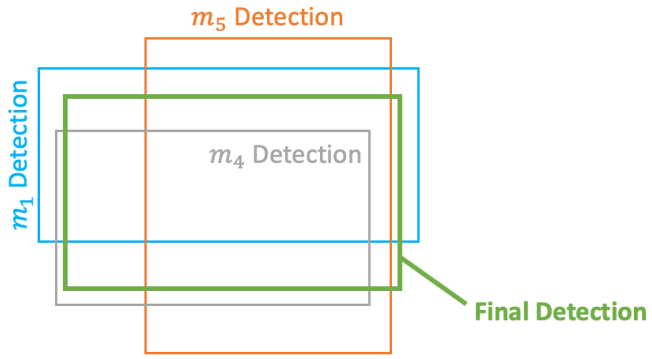

Fig. 3: Illustration of the final ensemble detection by weighted averaging of bounding box points with detection confidence. Detections are given by models m_1, m_4, and m_5. As m_1 is the *root* it is given higher priority in the weighted averaging and the green box is the final detection that will be yielded from this stack.

boxes to each other.

Given all the final stacks we compute their respective summed confidence scores. If this aggregated score exceeds a voting threshold of 1.68 (corresponding to an average score of 0.24 per model for $M = 7$), a final detection will be yielded from that stack (Fig. 3). The value 1.68 was found to optimize the trade-off between high mAP and high IoU. For each model we only considered detections with a confidence score greater than 0.2. Considering detections with a lower confidence score did not increase the performance but slowed down our ensemble method. The four corner points of the detection was then calculated according to a weighted (by the respective scores) average of all the bounding boxes from that stack. In order to reward the fact that a detection was confirmed by many models, we introduced a *frequency factor* of 0.03 that is multiplied by the number of boxes in that group and then added to the average score of the final detection. This added a slight improvement to our scores.

By visual inspection of the detection output we noticed that many bounding boxes were drawn around smaller bounding boxes of the same class label. As we deemed the smaller boxes to be superfluous, we added a post-processing step that removed final detections if another detection of the same class label with intersection-over-area (i.e. the ratio of intersection with a given box and its own ratio) $> 90\%$ was present. Thus, whenever a bounding was present and its area was more than 90% within another box of the same label, we removed the outer box. This improved our scores.

Another thing we observed is that there are frequently two or more detections of a different class that seem to overlap almost perfectly. In these scenarios we attempted to improve our score by removing the detection that had a lower confidence score, but this did not improve our score.

2.4. Single Models

We found that our test scores were optimal if we used seven models in our ensemble method. These seven models were selected and designed with the aim of achieving high dissimilarity between the models and high individual performance. Specifically we created the seven single models using different data augmentation techniques, different CNN depths and different configurations of the loss function.

Unless otherwise specified, all of these models used the 50-layer version of ResNet or ResNet-50 as the feature extractor. Our ResNet-50 was not trained from scratch but uses weights pre-trained on the MS COCO dataset. Initially we used a version with pre-trained weights from the ImageNet1k dataset, but experiments showed that MS COCO weights yield a better result (Table. 4). The deeper backbone networks, ResNet-101 or ResNet-152 used in two of the seven models in the ensemble, were pre-trained on ImageNet1k. Unless stated otherwise, training batch size of 1 was used. The number of training iterations used differ between models and is generally derived from the validation scores and with the goal of increasing the diversity between the models.

In the following we provide the specifications of the seven different models used in our ensemble method. The models are denoted m_1 to m_7 in descending order according to their single model performance, Table 5.

m_1 was our best performing model, where all configurations were optimized to the best of our knowledge. Besides changing the focal loss parameter γ from 2 to 1.5, thereby slightly reducing the extent to which the loss function prioritized hard examples, other parameters were mostly set as specified in [2]. Baseline data augmentation consisted of a randomized combination of image rotation, translation, shear, scaling and flipping. For each training epoch, each image was rotated, translated, and sheared by a factor of -0.1 to 0.1, scaled between 0.9 to 1.1 of its original size, and flipped with a chance of 50% both horizontally and vertically. m_1 trained for around 35k iterations.

Next, we tried to experiment with how the classification and regression loss were combined to yield the overall loss. By either increasing the weight of classification or regression loss we aimed to shift the focus between both losses. Doubling the weight of the classification loss provided a good balance between high performance and dissimilarity and hence this weighting was introduced for m_2. This model was trained for 81k iterations.

For the third model m_3 we chosen a different γ in the focal loss. We wanted this model to focus more on harder examples and set $\gamma = 3.5$ to achieve this. If one box was classified with a confidence score of 0.9, this means this box would contribute 100 times less to the loss in m_3 than in m_1. The model was trained for 58k iterations.

In the dataset we observed that the endoscopic frames were exposed to different illumination modes leading to dif-

ferent colorings of the images. Hence, we added a data augmentation step that randomly adds values to the RGB channels for model m_4. This was done in addition to the random geometric transformation applied in training all our models. For each epoch, there is a 1/9 chance for each image that a value between 50 and 200 was added to either of the RGB image channels. Training batch size was set to 4 and conducted for 10k iterations.

Model m_5 was trained with a 101-layer ResNet backbone. While this model performed worse than ResNet-50 models, we added it to the ensemble under the assumption that deeper CNNs will discover more advanced features and therefore add to the diversity of the ensemble. The model was trained for 45k iterations.

Analogously to m_5, we also added a model m_6 with a 152-layer ResNet as the feature extractor. m_6 was trained for 69k iterations.

The last model m_7 was trained on a subsampled trainset. For this model we added Gaussian noise at a scale of 127.5 (note 8-bit image intensity values of 0-255). Analogous to m_4, this augmentation step was added on top of the random geometric transformation and was applied to 1/9 of the images at each epoch. The model was trained using a batch size of 4 for 11k iterations.

3. EXPERIMENTS AND RESULTS

3.1. Dataset

Our dataset consists of the 2,193 endoscopic frames that were released by the EAD2019 challenge [3, 4]. A significant number of the frames in this dataset appear to be from the same video sequences. Further, these videos differed by tissue type, illumination mode and procedure type. In order to make sure that our train-validation split led to representative results, we had to make sure to split the dataset in a video-wise manner, meaning that one video was either entirely in the train set or entirely in the validation set based on manual assignment of the the frames to videos. Initial experiments conducted on a random train-validation split that did not respect a video-wise split resulted in validation scores up to 50% greater than the actual test scores submitted online. Our final validation set approximately correspond to 20% of the total EAD released training data.

3.2. Training

Models were trained with the Adam optimization algorithm [8]. We used a learning rate of 10^{-5} that was reduced by a factor of 10 whenever performance plateaued. Best performance were obtained for a training batch size of 1. Training was performed on a single GPU (Tesla K80) using Google Colab. Most runs were trained for 10 to 30 epochs (equal to 18k to 54k iterations) and took less 12 hours.

IoU Threshold	Validation Scores
0.25	0.2188
0.35	0.2402
0.4-0.5	**0.2861**
0.5	0.2669
0.6	0.2132
0.7	0.2531
0.8	0.1725

Table 1: Comparison of different IoU threshold for anchor assignment. All models trained for 12 epochs.

Backbone	Validation Score
VGG16	0.1181
VGG19	0.1305
ResNet50	**0.3165**
ResNet101	0.2879

Table 2: Comparison of backbone networks. VGG16 and VGG19 trained for less than 10 epochs as they stop improving before that.

3.3. Evaluation

Scores were calculated using a weighted sum of average IoU and mAP. IoU was weighted by 0.4 and mAP by 0.6. In order to avoid overly rewarding high IoU, the IoU value was additionally not allowed to exceed a multiple of 1.3 of the mAP. Otherwise it would have been possible for example to reach an overall score of 0.4 by having only one detection in the whole test set that overlapped perfectly with a ground-truth annotation.

3.4. Results

3.4.1. IoU Threshold

During training of the RetinaNet framework, anchors were considered a true positive based on the correctness of the predicted class and their IoU with the corresponding ground-truth annotation. We experimented with using different IoU thresholds to consider an anchor as true or false, Table. 1. We found that a negative threshold of 0.4 and a positive threshold of 0.5 worked best. This means that anchors with IoU below 0.4 were considered as false, above 0.5 were considered true, and those in between were ignored.

3.4.2. Backbone Network

As previously stated, we tried out different CNNs to use as the feature extractor in our RetinaNet framework. Even without pre-training on MS COCO, ResNet50 was the most effective and outperformed deeper ResNet models.

γ	**Validation Score**
1.00	0.2832
1.25	0.2915
1.50	**0.3235**
1.75	0.2905
2.00	0.3028
2.50	0.2780

Table 3: Validation scores for different γ values of our baseline model after 15 epochs. For values of γ below 1 RetinaNet failed to converge.

Pre-training	**Validation Score**
ImageNet1k	0.3108
MS COCO	**0.3435**

Table 4: Validation scores for different pre-trained weights of a ResNet-50 backbone network.

3.4.3. Focal Loss Parameters

Tuning the focal loss parameters had the greatest effect on our single model performance. Indeed, increasing or decreasing γ enabled us to decide to what extent we wanted the model to focus on hard examples. This was especially useful in our use case as the data was both unbalanced and some of the classes were much easier to detect than others. Using a γ value of 1.5 yielded the best performance for us. In the original paper [2] $\gamma = 2$ was used.

3.4.4. Pre-training

As previously mentioned, using a ResNet-50 model pre-trained on MS COCO improved our performance substantially compared to models pre-trained on Image1kNet.

3.4.5. Single Model Summary and Ensemble Method

The single model performance is summarised in Table. 5. We initially used 3 models in our ensemble method. By continuously adding models to the ensemble, our score kept on improving until we reached 7 models. Thereafter the score decreased again. Our performance was partly increased by an optimized combination strategy. This was largely thanks to the introduction of the *overlap score*, which handled the way boxes were assigned to each other, and to the *frequency factor* used to compute a weighted average of bounding box positions. The post-processing step of removing boxes that encompass boxes of the same class provided an additional performance boost. Finally, through testing and optimizing various parameters of our ensemble method we reached our final, highest score. These parameters include the following: *overlap score* threshold, the weighting between IoU and average score in the *overlap score*, *frequency factor*, score threshold of each individual model, and the overall voting threshold for each detection stack. Table. 6 summarise the describe stepwise improvement in test score. Our proposed ensemble method achieved a final score of 0.3451 on the EAD2019 test set. Our mAP was 0.3087 and IoU was 0.3997. For this submission, mAP on the EAD2019 generalization set was 0.2848 with a deviation score of 0.0696. In a previous submission, with slightly different ensemble parameters and the introduction of class-specific voting thresholds, we scored 33.45 on the test set and a mAP of 0.3508 on the generalization set with a deviation score of 0.0556.

Model	**Test Score**
m_1	0.3056
m_2	0.3033
m_3	0.2901
m_4	0.2856
m_5	0.2789
m_6	0.2750
m_7	0.2601

Table 5: The single model performances of the seven models used in our ensemble method.

Action	**Test** Score
Initial 3 Models	30.51
4 Models	31.93
5 Models	32.63
6 Models	32.95
7 Models	33.03
+Optimized Combination Strategy	33.88
+Post-Processing	33.96
+Parameter Optimization	**34.51**

Table 6: Summary of how different refinement steps led to score improvements towards our final ensemble method.

3.4.6. Visualization

Fig. 4-7 depict outputs from our ensemble method and models m_1, m_4 and m_5, respectively, on the same example image from the EAD2019 test set. The figures illustrate how different RetinaNet models are combined to produce a superior output.

4. DISCUSSION AND FUTURE WORK

Our approach tackles the novel issue of multi-class artefact detection in endoscopy by proposing the application of the one-stage detection method RetinaNet. RetinaNet matches the speed of other one-stage methods and its focal loss addresses the imbalance between easy and difficult examples.

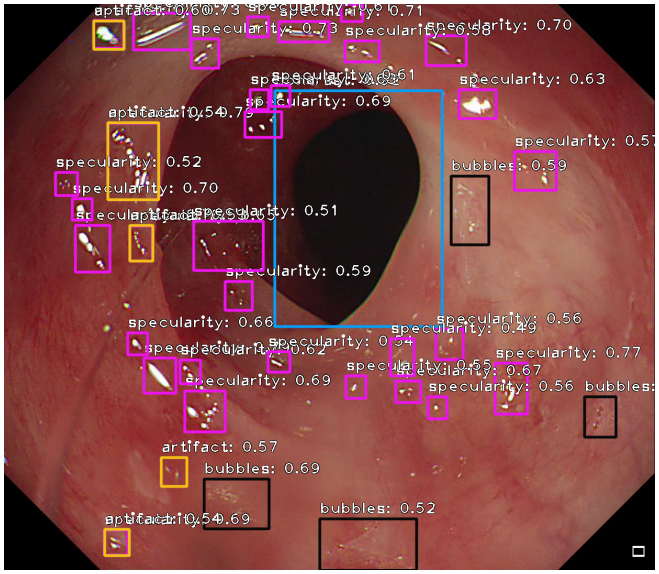

Fig. 4: Example output of combined ensemble method.

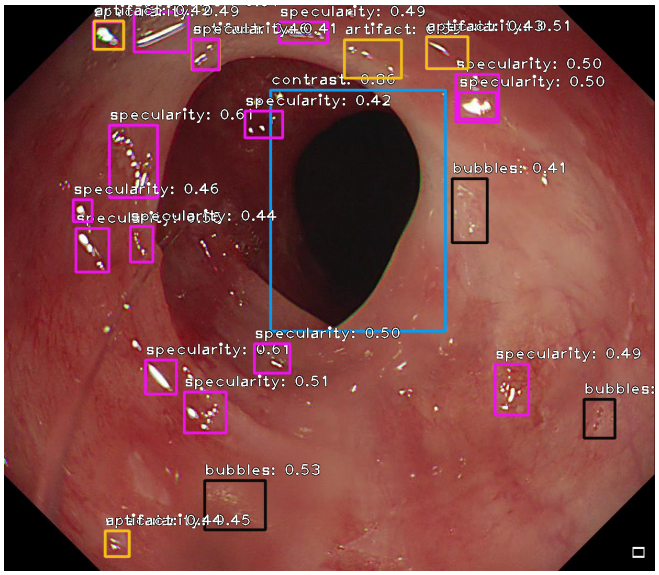

Fig. 5: Example output of our baseline model m_1.

By intelligently combining multiple models that were trained according to the specific nature of endoscopic video frames, our score improved substantially and resulted in a EAD2019 object detection score of 0.3451.

Future exploration could include the implementation of more advanced backbone networks and/or more advanced transfer learning approaches, such as pre-training on medical images.

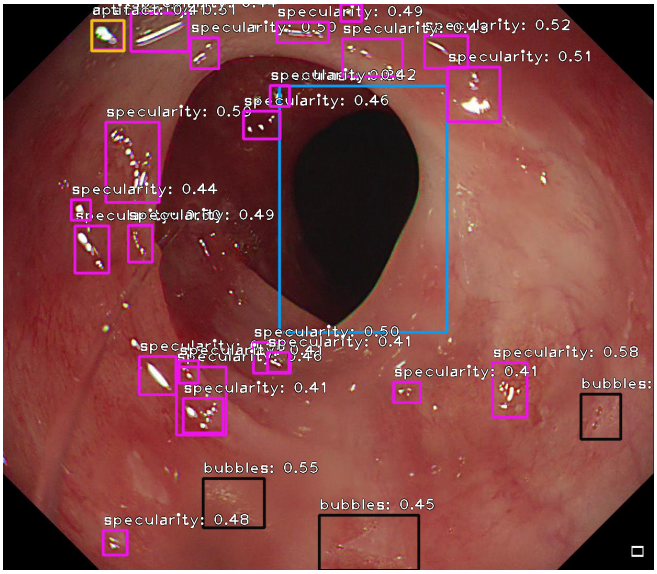

Fig. 6: Example output of color augmentation model m_4.

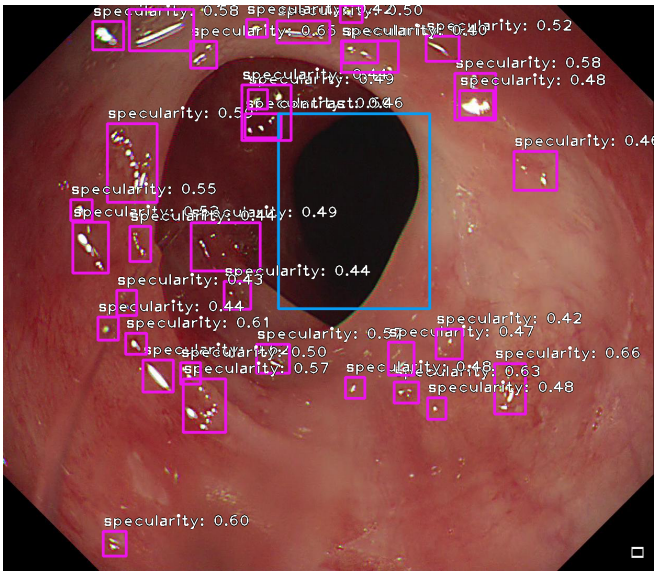

Fig. 7: Example output of ResNet-101 model m_5.

5. REFERENCES

[1] Pu Wang, Xiao Xiao, Jeremy R Glissen Brown, Tyler M Berzin, Mengtian Tu, Fei Xiong, Xiao Hu, Peixi Liu, Yan Song, Di Zhang, et al., "Development and validation of a deep-learning algorithm for the detection of polyps during colonoscopy," *Nature Biomedical Engineering*, vol. 2, no. 10, pp. 741, 2018.

[2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[3] Sharib Ali, Felix Zhou, Christian Daul, Barbara Braden, Adam Bailey, Stefano Realdon, James East, Georges Wagnires, Victor Loschenov, Enrico Grisan, Walter Blondel, and Jens Rittscher, "Endoscopy artifact detection (EAD 2019) challenge dataset," *CoRR*, vol. abs/1905.03209, 2019.

[4] Sharib Ali, Felix Zhou, Adam Bailey, Barbara Braden, James East, Xin Lu, and Jens Rittscher, "A deep learning framework for quality assessment and restoration in video endoscopy," *CoRR*, vol. abs/1904.07073, 2019.

[5] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.

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

[7] 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*, 2016, pp. 770–778.

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