Tradeoffs using Binary and Multiclass Neural Network Classification for Medical Multidisease Detection

Tor Jan Derek Berstad[1,3], Michael Alexander Riegler[1,3]
Konstantin Pogorelov[2,3], Håkon Kvale Stensland[2,3], Pål Halvorsen[1,3]
[1]*Simula Metropolitan Center for Digital Engineering, Norway*
[2]*Simula Research Laboratory, Norway*
[3]*University of Oslo, Norway*

***Abstract*—The interest in neural networks has increased significantly, and the application of this type of machine learning is vast, ranging from natural image classification to medical image segmentation. However, many users of neural networks tend to use them as a black box tool. They do not access all of the possible variations, nor take into account the respective classification accuracies and costs. In our work, we focus on multiclass image classification, and in this research, we shed light on the trade-offs between systems using a single multiclass classification and multiple binary classifiers, respectively. We have tested the these classifiers on several modern neural network architectures, including DenseNet, Inception v3, Inception ResNet v2, Xception, NASNet and MobileNet. We have compared several aspects of the performance of these architectures during training and testing using both classification styles. We have compared classification speed and several classification accuracy metrics. Here, we present the results from experiments on a total of 99 networks: 11 multiclass and 88 individual binary networks, for an 8-class classification of medical images. In short, using multiple binary classification networks resulted in a 7% increase in the average F1 score, a 1% increase in average accuracy, a 1% increase in precision, and a 4% increase in average recall. However, on average, such a multi-network style performed the classification 7.6 times slower compared to a single network multiclass implementation. These collective findings show that both approaches can be applied to modern neural network structures. Several binary networks will often give increased classification accuracy, but at the cost of classification speed and resource consumption.**

***Index Terms*—Multiclass classification, accuracy, system performance**

I. Introduction

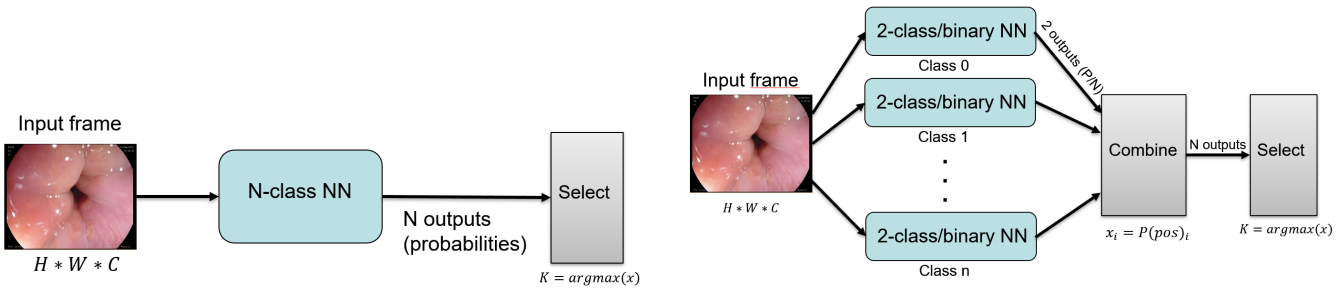
In recent years, interest in and research on machine learning have skyrocketed [4]. There are many potential applications of machine learning, with researchers constantly envisioning new applications and solutions. One of the most promising applications of many types of machine learning is in the area of analyzing images. The hope is that computers will be able to assist humans in many types of image-related task such as classification, segmentation and 3D reconstruction.

The most common task is classification of images, which presents the challenge of finding the best approach for a given classification problem. Many use neural networks as a black box tool and are concerned only with accuracy. However, the problem is complex as there are multiple approaches, and accuracy is just one part of the entire system performance. In our research, we try to balance the performance of the entire system and evaluate the trade-offs of various solutions in a multi-class detection scenario [13], [15], [16]. In this paper, as shown in figure 1, we compare the performance of multiple binary classifiers, also known as one-vs-the-rest (OvR), to recently used multiclass classifiers in a (medical) multiclass classification scenario. The OvR approach means that we have one classifier per class and combine the output of all of the binary classifiers to create a multiclass classifier.

Thus, the open question is whether it is beneficial to apply and combine the OvR technique with several binary classifiers into one classifier, or use a large complex multiclass classifier. Our assumption is that the binary classifiers could become better at learning the specific features of the class in question, and that the classification performance of the combined classifier is better than one large multiclass system. However, we also believe that these single-class binary classifiers could be more robust against poor hyperparameter selection and un-optimized network structures. Nevertheless, running multiple instances of a classifier may require more system resources. To the best of our knowledge, very little research has been undertaken to investigate these trade-offs. Whenever OvR has been applied, it has tended to use more traditional binary classifiers that support regularization, e.g., support vector machines (SVMs), where the OvR approach seems to generate promising results for some multiclass networks [8] [17] [19]. However, we are not aware of any application of OvR for common advanced neural networks or comparisons.

As a multiclass classification problem, we use disease detection in the gastrointestinal (GI) tract as a case study, for which the classifier should detect images (or video frames) that contain specific GI landmarks or anomalies. In this area, a significant challenge is that important anomalies are missed [9] and there are large inter- and intra-clinician variations [14], [18].

The aim is to develop a computer-assisted diagnosis tool that can be of benefit in endoscopic examinations. In this type of real-life system, the entire system performance is important: i)



(a) Conventional neural network classifier

(b) OvR neural network classifier

Fig. 1: Different classifier setups used, conventional neural network and OvR ensemble

high accuracy is required to classify both normal and abnormal tissues correctly; and ii) the resource consumption on the computers used should be low in order to enable real-time diagnosis feedback to the medical experts, as well as saving resources in terms of both cost and scalability.

In this paper, we use an 8-class medical dataset [12] called Kvasir to shed light on the trade-offs between a single multi-class classification system versus multiple binary classifiers (OvR). We have researched transfer learning using several modern classifiers of well-known neural network architectures, i.e., DenseNet, Inception v3, Inception ResNet v2, Xception, NASNet and MobileNet. We have compared several aspects of their performance during training and testing using both classification methods. We have analyzed resource consumption, classification speed, and several classification accuracy metrics. We have trained 186 different networks, resulting in 16,029,806 metric data points relating to both classification and system performance. In particular, we present the results from a total of 99 networks, i.e., 11 multiclass and 88 individual binary networks, for an 8-class classification of medical images.

In summary, our experimental results show that the OvR style does provide satisfactory classification performance at the expense of higher resource use and slower classification speed. We observed a 7% increase in the average F1 score, a 1% increase in average accuracy, a 6% increase in Matthews correlation coefficient (MCC), a 1% increase in precision, and a 4% increase in average recall. Specificity remained relatively unchanged. In addition, the median values for all of these metrics increased significantly in the OvR style, with the median F1, MCC and recall scores increasing by over 15%. The results from the most improved network in the OvR configuration indicated an increase in F1 of 45% and an MCC increase of 40%. However, on average, running several binary networks was 7.6 times slower than a single multiclass network implementation. These collective findings show that both approaches can be applied to modern neural network structures. Several binary networks will often give increased classification accuracy, but at the cost of classification speed and resource consumption.

II. Methodology

We decided to benchmark 11 neural network architectures ending in a total of 99 tested network configurations. Given that it can take up to 29 hours to train a single network, as we have configured them, this would not have been possible without a specific framework as it would have been too time-consuming. Therefore, we developed an open-source test suite[1]. Furthermore, to run the necessary experiments, we used a system configuration with an Intel Core i5-4590, 16GB memory and a single NVIDIA GeForce GTX 1080 Ti GPU. The system should be capable of adequate performance for our use, in addition to being representative of modern high-performance hardware. Our GPU had compute compatibility of 6.1 [6], which was the highest available at the time of writing for a consumer GPU. Note that the system is only equipped with a single GPU, while TensorFlow is able to use several GPUs in parallel [3].

To evaluate the system, we used the Kvasir medical image dataset [12], which contains 8,000 images in total. We chose to divide the images into three sets using a seeded random process, which creates symlinks to existing data in a pseudo-random manner. The training set is used for training directly. The validation set is used for the validation phase of training, in which the validation accuracy is calculated and is used primarily for our early-stopping function. Finally, we have the test set, which is not used for training at all and is not seen by the network until the final testing phase. This process uses *numpy*'s random selection. We selected a 50-25-25 split, as suggested by Marsland [10]. Thus, we had 4,000 training images, 2,000 validation images and 2,000 test images. The images were selected proportionally per class, i.e., each set had the same number of images for each class.

III. Experiments

In order to properly assess the final classification performance of both binary and multiclass networks, we chose a selection of metrics that together created a decent summary. Some of these metrics are better for assessing the performance of multiclass networks, and some are better for binary problems. If we look at the multiclass problem as a series of

[1] https://github.com/Berstad/TFmetrics

binary problems, both styles can be helpful. The metrics are based primarily on Marsland's book [10], and are also defined in [12]: true positive (TP), true negative (TN), false positive (FP), false negative (FN), recall (REC, also frequently called sensitivity), precision (PREC), specificity (SPEC), accuracy (ACC), Matthews correlation coefficient (MCC), F1 score (F1) and frames per second (FPS). The metrics are then macro-averaged for each class, for both the OvR networks and the multiclass network. Macro-averaging was deemed acceptable because our classes were balanced for the test data.

As our trained networks have the potential for use in a clinical setting [12], it is helpful for us to define a threshold for all, or at least several, of our metrics that show us whether or not the resulting classification is suitable for diagnostic work. The threshold for a network becoming a good classifier is difficult to define precisely, as such thresholds vary based on the data in question and the application. Moreover, some metrics might be more important than others in a medical scenario, and particularly metrics that place a high value on avoiding false negatives. Metrics in which false negatives heavily reduce the score are more suitable to optimize. One such metric is recall or sensitivity. Research suggests that a reasonable threshold for a network to be used for diagnostic purposes is for recall to be greater than 0.85 [11], and specificity should be higher than 0.85.

In addition, we set the goal that our networks should be able to process data in real time. This means that they should be able to process frames at least as quickly as the equipment is able to produce them. After searching for several different types of endoscopic equipment, we found that the number of frames per second varied considerably. Some wireless pill-type cameras produced as few as 3 FPS [2], whereas high-end wired equipment produced up to 60 FPS [1]. Thus, the higher frame rate is important to support high-end cameras, but some standard medical equipment feature a 30 FPS output [7], so we use this as the minimum FPS threshold.

Table I presents a summary of the chosen built-in architectures of Keras, detailing the number of parameters, the number of layers, the depth, and the size on disk. There is a large variety of complexities and network sizes. Interestingly, this shows one of the first issues with a binary implementation, which requires one fully trained model per class. The number of classes and the network size mean that, in the case of NASNet Large, for example, we require approximately 8 GB of storage for eight networks. These models must be stored on disk and, perhaps even more importantly, must be loaded into video memory during the classification operations. The large size of these models poses a challenge for systems in which video memory is limited.

Keras offers us the ability to import networks with pre-trained weights; in other words, this is an easy way to implement transfer learning. We decided to use this ability and, as our problem is an excellent example of image classification based on features, we decided to use available networks that were already trained on the ImageNet dataset. This allows us to train our networks much faster and more efficient.

Network	Parameters (millions)	Depth	Number of Layers	Size on disk
VGG16	138.35 [5]	21	21	115.6 MB
VGG19	143.6 [5]	24	24	155.7 MB
Inception v3	23.85 [5]	159	313	177.1 MB
DenseNet 121	8 [5]	121	428	81.3MB
DenseNet 169	14.3 [5]	169	596	69.0 MB
DenseNet 201	20.242 [5]	201	708	94.4 MB
Xception	22.9 [5]	126	134	121.9 MB
Inception RN v2	55.87 [5]	572	782	261.3 MB
MobileNet	4.25 [5]	88	98	32.4 MB
NASNet Large	88.9 [20]	768	1021	1002.0 MB
NASNet Mobile	5.3 [20]	384	751	54.8 MB

TABLE I: The chosen architectures with specifications, depth and layers adjusted to match our actual implementation. Size is based on the size of trained models.

In our implementation, we do not include the top of the pre-defined networks and, instead, build our own top layers: a pooling layer and a fully connected layer with two or eight outputs, depending on the configuration. Then, during the initial training, we freeze all weights other than those in our top layers. This allows us to quickly gain a degree of accuracy while retaining much of the pre-trained information. After this, we unfreeze a certain number of blocks at the top of the imported network and "fine tune" with a lower learning rate. Unfreezing more of the weights allows the network to learn more about the data.

Table II presents the selected hyperparameters for each architecture. As we can observe, the hyperparameters for the binary networks (our OvR structure) are the same as for our multiclass architectures. Hyperparameters were chosen based on tests of different combinations using grid search.

	Network	BS	Image dimensions	Optimizer	LR (Train)	LR (Tune)	BMLBLN	Epochs (Max)
MULTI	VGG16	64	224x224	Nadam	0.002	1e-06	15	200
	VGG19	64	224x224	Nadam	0.002	1e-06	17	200
	Inception v3	64	299x299	Nadam	0.002	1e-05	249	200
	DenseNet 121	64	224x224	Nadam	1e-04	1e-05	394	200
	DenseNet 169	16	224x224	Nadam	1e-04	1e-05	530	200
	DenseNet 201	64	224x224	Nadam	1e-04	1e-06	642	200
	Xception	16	299x299	Nadam	0.002	1e-05	126	200
	Inception RN v2	64	299x299	Nadam	0.002	1e-05	64	200
	MobileNet	64	224x224	Nadam	1e-03	1e-04	762	200
	NASNet Large	8	331x331	Nadam	1e-05	1e-06	100	100
	NASNet Mobile	64	224x224	Nadam	1e-05	1e-06	0	200
BINARY	VGG16	64	224x224	Nadam	0.002	1e-06	15	25
	VGG19	64	224x224	Nadam	0.002	1e-06	17	25
	Inception v3	64	299x299	Nadam	0.002	1e-05	249	25
	DenseNet 121	64	224x224	Nadam	1e-04	1e-05	394	25
	DenseNet 169	16	224x224	Nadam	1e-04	1e-05	530	25
	DenseNet 201	64	224x224	Nadam	1e-04	1e-06	642	25
	Xception	16	299x299	Nadam	0.002	1e-05	126	25
	Inception RN v2	64	299x299	Nadam	0.002	1e-05	64	25
	MobileNet	64	224x224	Nadam	1e-03	1e-04	762	25
	NASNet Large	8	331x331	Nadam	1e-05	1e-06	100	25
	NASNet Mobile	64	224x224	Nadam	1e-05	1e-06	0	25

TABLE II: Chosen hyperparameters for each network including batch size (BS), image dimantions, optimizer, learning rate (LR), based model last block layer number (BMLBLN), and number of training epochs.

IV. Classification performance and speed discussion

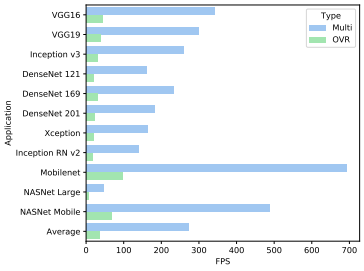
In this section, we will look at how all the networks performed during classification. Of particular importance is how quickly each style and architecture was able to process and classify frames. In addition, the classification accuracy shows how OvR performs regarding overall accuracy, even

when the hyperparameters are not optimized for the individual binary networks.

In total, we collected data on the training, fine tuning and testing sessions of 186 networks. Fifty of these were multiclass and 136 were individual binary networks. In our final test selection of 99 networks (11 multiclass and 88 binary), the metrics logged and calculated resulted in 16,029,806 total data points collected.

A. System Performance

To test the frame throughput, or FPS, of each network, we ran an experiment using the 2,000 frames in our test set. In the OvR case, this mean that the frames were run through the eight networks in parallel. We observed a slight variance in FPS and decided to run the test ten times sequentially to correct for this. Effectively, we were testing our network on its ability to process 20,000 frames quickly, which is equivalent to approximately 11.1 minutes of video with a framerate of 30 FPS. We did, however, log the FPS achieved for each 2,000-frame test. The summary of these tests can be seen in Figure 2.



(a) FPS bar plot.

Application	Multi	OvR	SDF
VGG16	341.62	44.49	7.68x
VGG19	299.49	38.52	7.77x
Inception v3	260.18	31.33	8.30x
DenseNet 121	160.89	20.67	7.78x
DenseNet 169	232.91	29.97	7.77x
DenseNet 201	183.31	23.59	7.77x
Xception	163.27	21.08	7.75x
Inception RN v2	139.86	17.72	7.89x
MobileNet	692.59	95.79	7.23x
NASNet Large	47.56	6.03	7.89x
NASNet Mobile	486.83	66.75	7.29x
Average	273.50	35.99	7.60x

(b) Raw FPS averages.

Fig. 2: FPS test summary for all networks, including all 10 tests and average value. The slowdown factor (SDF) shows how many times slower the OvR network is than the multiclass equivalent. From the averages, we can see that using single multiclass networks is approximately 7.6 times faster than using the OvR style.

From the figure, we clearly see that the OvR network style has a notable negative impact on frame throughput. Every architecture is consistently faster in the multiclass configuration; this is not surprising given the resource allocation characteristics we observed in the previous chapters. We can see that the slowdown experienced, on average, is almost proportional to the number of classes. We interpret this to mean that our binary networks are saturating what little resources are left over from a single multiclass network, and all of the additional resources required are causing the performance to decrease proportionally. This means that, for a dataset with more classes, such as ImageNet with over 20,000 classes, it would not be an appropriate strategy as the time needed to perform the analyzis would be too long.

However, there are two interesting points to note here. As we can see, the slowdown factor (SDF) is higher for Inception v3. The delay reduces the average FPS rather dramatically for this architecture in the OvR style. In addition, we note that the SDF appears to be lessened for the two high-efficiency architectures (MobileNet and NASNet Mobile). This is an expected outcome, but it shows the effect of their initial resource usage not being as high as the other architectures in the multiclass style. We can also see that our fastest networks, both MobileNet, experienced the least amount of relative slowdown of all our architectures. This confirm that, if a network is using fewer resources in a typical configuration, the consequences of using it in an OvR configuration will be lessened, at least from a performance standpoint.

We can also see from Figure 2 that there is a significant difference in the architectures in terms of frame throughput. Our fastest architecture, MobileNet, is over ten times faster than our slowest, NASNet Large. In the tools and implementation section, we detailed our goals for the achieved FPS to enable real-time performance. We set a minimum FPS of 30 and a goal of 60. From the numbers, we can see that all but one of our architectures achieved 60 FPS or higher in the multiclass configuration, and even this architecture achieved at least 30 FPS.

However, considering OvR, we can see that the numbers are not as promising. Only two of our chosen architectures were able to achieve above 60 FPS; unsurprisingly, these were the two high-efficiency architectures. A further three architectures were able to achieve over 30 FPS, which means that six of our 11 architectures fell below our minimum 30 FPS goal, and some by quite substantial amounts. Thus, the current networks are not likely to be suitable for real-time operation. The most substantial feasibility decrease was for NASNet Large, which achieved only 6 FPS.

On the other hand, MobileNet was still able to achieve 95 FPS in the OvR configuration, which is surprisingly useful and almost twice as fast as the slowest multiclass classifier. To summarize, there is a substantial performance penalty associated with choosing an OvR style. These penalties are so substantial that they will render some of our networks unable to classify in real-time. The choice of architecture is also critical, and a good takeaway from this section is not to choose an architecture that is more complex and resource hungry than required.

To determine the effect that network complexity has on performance, we can examine the same table we presented in the tools and implementation section, but with the FPS values from our tests included. In Table III, the basic network complexity details and the achieved FPS are presented. However, the table does not tell the whole story as it does not include the not detail exactly which layer types are chosen in the architecture and how they are connected. However, it is still interesting to examine the relationships between the different specifications and their performance.

We can see, for example, that the VGG architectures have an enormous number of trainable parameters, by far the most of the architectures we tested. However, they are also the two shallowest networks. Both VGG networks achieved above average performance (>273.5 FPS). The DenseNet architectures

Network	Parameters (M)	Depth	Layers	Size	FPS (Multi)	FPS (OvR)
VGG16	138.35 [5]	21	21	115.6MB	341.62	44.49
VGG19	143.6 [5]	24	24	155.7MB	299.49	38.52
Inception v3	23.85 [5]	159	313	177.1MB	260.18	31.33
DenseNet 121	8 [5]	121	428	81.3MB	160.89	20.67
DenseNet 169	14.3 [5]	169	596	69MB	232.91	29.97
DenseNet 201	20.242 [5]	201	708	94.4MB	183.31	23.59
Xception	22.9 [5]	126	134	121.9MB	163.27	21.08
Inception RN v2	55.87 [5]	572	782	261.3MB	139.86	17.72
MobileNet	4.25 [5]	88	98	32.4MB	692.59	95.79
NASNet Large	88.9 [20]	768	1021	1002MB	47.56	6.03
NASNet Mobile	5.3 [20]	384	751	54.8MB	486.83	66.75

TABLE III: A list of the chosen architectures with specifications, depth and layers adjusted to match our actual implementation, in addition to the achieved FPS in our tests. Size is based on actual trained models.

have far fewer trainable parameters but achieved below average performance. The two fastest networks also have the fewest trainable parameters, but it seems evident that the number of trainable parameters does not provide full details of the performance.

Network depth does not seem to affect the achieved FPS either. In Table III, we can see that NASNet mobile is the third-deepest network, but the second fastest. Conversely, DenseNet 121 is the fourth-shallowest network, yet also the third slowest. In fact, after attempting regression analysis on the test runs for each architecture, none of these specifications seemed to correlate with the total performance. It is more likely that the types of layer, block and connection have an effect on the total performance. In future research, a more in-depth analysis of the relationship should be carried out.

B. Classification Accuracy

While the frame throughput is important, it is of little consequence if the resulting classification accuracy does not meet our expectations. Thus, we need to determine how well, or poorly, our networks were able to classify the test data.

	Network style	FN	FP	TN	TP	F1	ACC	MCC	PREC	REC	SPEC
Multi	VGG16	35.38	35.38	1714.62	214.62	0.86	0.96	0.84	0.86	0.86	0.98
	VGG19	35.50	35.50	1714.50	214.50	0.86	0.96	0.84	0.86	0.86	0.98
	Inception v3	81.50	81.50	1668.50	168.50	0.62	0.92	0.63	0.77	0.67	0.95
	DenseNet 121	113.50	113.50	1636.50	136.50	0.50	0.89	0.52	0.76	0.55	0.94
	DenseNet 169	111.88	111.88	1638.12	138.12	0.51	0.89	0.51	0.69	0.55	0.94
	DenseNet 201	148.12	148.12	1601.88	101.88	0.34	0.85	0.35	0.66	0.41	0.92
	Xception	82.12	82.12	1667.88	167.88	0.65	0.92	0.63	0.72	0.67	0.95
	Inception RN v2	79.62	79.62	1670.38	170.38	0.66	0.92	0.64	0.72	0.68	0.95
	MobileNet	98.88	98.88	1651.12	151.12	0.56	0.90	0.55	0.74	0.60	0.94
	NASNet Large	42.50	42.50	1707.50	207.50	0.83	0.96	0.81	0.83	0.83	0.98
	NASNet Mobile	31.38	31.38	1718.62	218.62	0.87	0.97	0.86	0.88	0.87	0.98
	Average	78.22	78.22	1671.78	171.78	0.66	0.92	0.65	0.77	0.69	0.96
OvR	VGG16	46.62	46.62	1703.38	203.38	0.81	0.95	0.79	0.82	0.81	0.97
	VGG19	44.38	44.38	1705.62	205.62	0.82	0.96	0.80	0.83	0.82	0.97
	Inception v3	58.50	58.50	1691.50	191.50	0.76	0.94	0.74	0.79	0.77	0.97
	DenseNet 121	109.50	109.50	1640.50	140.50	0.53	0.89	0.53	0.69	0.56	0.94
	DenseNet 169	94.12	94.12	1655.88	155.88	0.60	0.91	0.59	0.71	0.62	0.95
	DenseNet 201	146.25	146.25	1603.75	103.75	0.39	0.85	0.39	0.65	0.41	0.92
	Xception	91.12	91.12	1658.88	158.88	0.65	0.91	0.63	0.77	0.64	0.95
	Inception RN v2	64.12	64.12	1685.88	185.88	0.74	0.94	0.72	0.79	0.74	0.96
	MobileNet	45.62	45.62	1704.38	204.38	0.81	0.95	0.80	0.86	0.82	0.97
	NASNet Large	33.00	33.00	1717.00	217.00	0.87	0.97	0.85	0.87	0.87	0.98
	NASNet Mobile	51.25	51.25	1698.75	198.75	0.79	0.95	0.77	0.80	0.80	0.97
	Average	71.32	71.32	1678.68	178.68	0.71	0.93	0.69	0.78	0.71	0.96
	Average Difference	-6.90	-6.90	+6.90	+6.90	+0.05	+0.01	+0.04	+0.01	+0.03	0.00

TABLE IV: Metric averages for all network styles and architectures.

In Table IV, we have created an extensive summary of how well our networks were able to classify the data. In the table, we have highlighted the best results achieved for each metric in both the multiclass and OvR styles. Of immediate interest here is that the best-performing multiclass architecture is not the best-performing OvR architecture. It is also interesting to note that the best-performing OvR architecture is very similar to the best multiclass in terms of classification performance, with only 1 TP and 1 TN difference between them on average. We can also see, with the help of the average difference fields, that the OvR classifiers perform slightly better than multiclass classifiers in total.

To see the changes for each metric for each architecture, Table V presents the differences from multiclass to OvR, making the differences between the OvR and multiclass architectures more apparent. For example, we can see that both VGG architectures performed worse in the OvR style, and by approximately the same amount. The Inception architectures performed better in OvR, as did NASNet Large. Xception performed slightly worse, and NASNet Mobile performed significantly worse, showing the most significant detriment to accuracy from switching to the OvR style.

The DenseNet styles performed better, on average, with OvR, except in terms of precision in the case of 121 and 201, where they performed worse. This seems odd and is discussed further, below. Finally, we can see that MobileNet saw a significant improvement in performance, from being one of the worst classifiers to becoming one of the better classifiers by switching from single-network multiclass to OvR.

Network style	FN	FP	TN	TP	F1	ACC	MCC	PREC	REC	SPEC
VGG 16	+11.25	+11.25	-11.25	-11.25	-0.05	-0.01	-0.05	-0.05	-0.05	-0.01
VGG 19	+8.88	+8.88	-8.88	-8.88	-0.04	-0.01	-0.04	-0.03	-0.04	-0.01
Inception v3	-23.00	-23.00	+23.00	+23.00	+0.14	+0.02	+0.11	+0.02	+0.09	+0.01
DenseNet 121	-4.00	-4.00	+4.00	+4.00	+0.03	0.00	+0.01	-0.07	+0.02	0.00
DenseNet 169	-17.75	-17.75	+17.75	+17.75	+0.09	+0.02	+0.08	+0.03	+0.07	+0.01
DenseNet 201	-1.88	-1.88	+1.88	+1.88	+0.05	0.00	+0.04	-0.02	+0.01	0.00
Xception	+9.00	+9.00	-9.00	-9.00	-0.00	-0.01	0.00	+0.05	-0.04	-0.01
Inception RN v2	-15.50	-15.50	+15.50	+15.50	+0.08	+0.02	+0.08	+0.07	+0.06	+0.01
MobileNet	-53.25	-53.25	+53.25	+53.25	+0.25	+0.05	+0.25	+0.11	+0.21	+0.03
NASNet Large	-9.50	-9.50	+9.50	+9.50	+0.04	+0.01	+0.04	+0.03	+0.04	+0.01
NASNet Mobile	+19.88	+19.88	-19.88	-19.88	-0.08	-0.02	-0.09	-0.08	-0.08	-0.01
Average Difference	-6.90	-6.90	+6.90	+6.90	+0.05	+0.01	+0.04	+0.01	+0.03	0.00
Average Difference (%)	-8.82	-8.82	+0.41	+4.02	+7.02	+0.79	+5.99	+1.06	+4.11	+0.38
Median Difference (%)	-28.22	-28.22	+1.38	+13.65	+16.92	+2.17	+17.46	+3.95	+14.93	+2.11

TABLE V: Metric average differences from multiclass to OvR for all network styles and architectures.

1) Failure analysis: To understand performance differences, we provide a closer examination of a case in which the OvR performed worse and another in which it performed better. As we can see, NASNet Mobile achieved the worst result using the OvR style. First, we examine the raw figures for each class, which are seen in Table VI. Reviewing the table, the problem is clear: every class is classified less accurately, almost irrespective of which metric we choose. There are a few small exceptions, such as class 5 having higher precision in the OvR style and class 2 having higher recall.

Inspecting the confusion matrix in Figure 3, we can see the effect more clearly. Our true positives, indicated on the diagonal, are fewer for each class, except class 2. Likewise, our false positives are higher for each class, except class 5. We also get an indication from this plot that classes 2 and 5

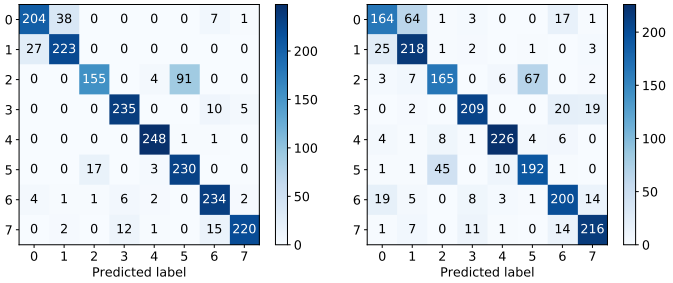
Style	Class	FN	FP	TN	TP	F1	ACC	MCC	PREC	REC	SPEC
MULTI	class 0	46	31	1719	204	0.84	0.96	0.82	0.87	0.82	0.98
	class 1	27	41	1709	223	0.87	0.97	0.85	0.84	0.89	0.98
	class 2	95	18	1732	155	0.73	0.94	0.72	0.90	0.62	0.99
	class 3	15	18	1732	235	0.93	0.98	0.92	0.93	0.94	0.99
	class 4	2	10	1740	248	0.98	0.99	0.97	0.96	0.99	0.99
	class 5	20	92	1658	230	0.80	0.94	0.78	0.71	0.92	0.95
	class 6	16	33	1717	234	0.91	0.98	0.89	0.88	0.94	0.98
	class 7	30	8	1742	220	0.92	0.98	0.91	0.96	0.88	1.00
	Average	31.38	31.38	1718.62	218.62	0.87	0.97	0.86	0.88	0.87	0.98
BINARY	class 0	86	53	1697	164	0.70	0.93	0.67	0.76	0.66	0.97
	class 1	32	87	1663	218	0.79	0.94	0.76	0.71	0.87	0.95
	class 2	85	55	1695	165	0.70	0.93	0.66	0.75	0.66	0.97
	class 3	41	25	1725	209	0.86	0.97	0.85	0.89	0.84	0.99
	class 4	24	20	1730	226	0.91	0.98	0.90	0.92	0.90	0.99
	class 5	58	73	1677	192	0.75	0.93	0.71	0.72	0.77	0.96
	class 6	50	58	1692	200	0.79	0.95	0.76	0.78	0.80	0.97
	class 7	34	39	1711	216	0.86	0.96	0.83	0.85	0.86	0.98
	Average	51.25	51.25	1698.75	198.75	0.79	0.95	0.77	0.80	0.80	0.97
	Average Diff.	+19.88	+19.88	-19.88	-19.88	-0.08	-0.02	-0.09	-0.08	-0.08	-0.01

TABLE VI: Accuracy for NASNet mobile.

Style	class	FN	FP	TN	TP	F1	ACC	MCC	PREC	REC	SPEC
MULTI	class 0	89	111	1639	161	0.62	0.90	0.56	0.59	0.64	0.94
	class 1	201	0	1750	49	0.33	0.90	0.42	1.00	0.20	1.00
	class 2	248	0	1750	2	0.02	0.88	0.08	1.00	0.01	1.00
	class 3	29	34	1716	221	0.88	0.97	0.86	0.87	0.88	0.98
	class 4	0	337	1413	250	0.60	0.83	0.59	0.43	1.00	0.81
	class 5	89	202	1548	161	0.53	0.85	0.45	0.44	0.64	0.88
	class 6	48	99	1651	202	0.73	0.93	0.70	0.67	0.81	0.94
	class 7	87	8	1742	163	0.77	0.95	0.77	0.95	0.65	1.00
	Average	98.88	98.88	1651.12	151.12	0.56	0.90	0.55	0.74	0.60	0.94
BINARY	class 0	9	104	1646	241	0.81	0.94	0.79	0.70	0.96	0.94
	class 1	125	0	1750	125	0.67	0.94	0.68	1.00	0.50	1.00
	class 2	128	5	1745	122	0.65	0.93	0.66	0.96	0.49	1.00
	class 3	17	15	1735	233	0.94	0.98	0.93	0.94	0.93	0.99
	class 4	0	117	1633	250	0.81	0.94	0.80	0.68	1.00	0.93
	class 5	40	107	1643	210	0.74	0.93	0.71	0.66	0.84	0.94
	class 6	26	10	1740	224	0.93	0.98	0.92	0.96	0.90	0.99
	class 7	20	7	1743	230	0.94	0.99	0.94	0.97	0.92	1.00
	Average	45.62	45.62	1704.38	204.38	0.81	0.95	0.80	0.86	0.82	0.97
	Average Diff.	-53.25	-53.25	+53.25	+53.25	+0.25	+0.05	+0.25	+0.11	+0.21	+0.03

TABLE VII: Accuracy for Mobilenet.

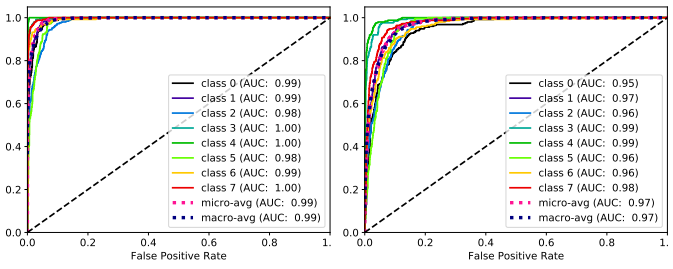
are often confused with each other, and the same with classes 0 and 1.



(a) Multiclass confusion matrix (b) Binary confusion matrix

Fig. 3: Confusion matrices for NASNet Mobile.

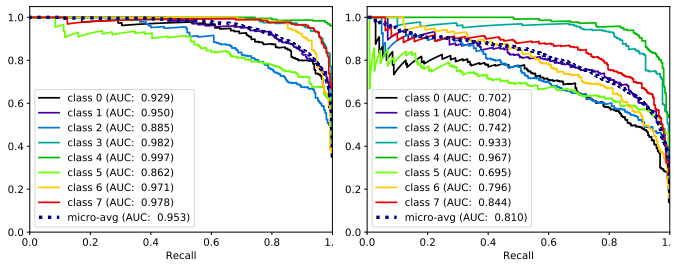
Looking at the Receiver Operating Characteristic (ROC) curves in Figure 4, we can see that the results are very good for the multiclass case, as several of the curves have an AOC ≥ 0.99. Both the micro- and macro-averaged AUC values are 0.99. Unsurprisingly, this is our best classifier in the multiclass configuration. In the OvR case, however, the results are diminished slightly. Specifically, classes 0, 2, 5 and 6 are a little worse. However, AUC values of 0.95 or higher are still usually an indication of functional classification, and we should also examine the PR curves.



(a) Multiclass ROC curves (b) Binary ROC curves

Fig. 4: ROC curves for NASNet Mobile.

From the PR curves, seen in Figure 5, the effect of our poorest classes is much more apparent. The difference between the multiclass and OvR styles is clearly visible. It also shows us a more realistic picture of how good the multiclass classifier is. The AUC is no longer almost perfect, and the differences between classes become more easily visible as well. Classes 2 and 5 can be seen here reducing the average AUC, and they are a fairly significant amount below average. Interestingly, although some of the values for classes 2 and 5 seem to improve in the main chart, we can see that they are also worse if we take into account both precision and recall.



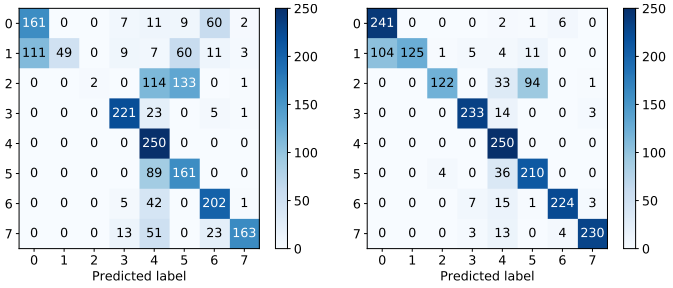
(a) Multiclass PR curves (b) Binary PR curves

Fig. 5: PR curves for NASNet Mobile.

In contrast to NASNet Mobile, MobileNet seems to perform much better in the OvR style. The results for the multiclass are quite weak (certainly below our desired goals), while the OvR style meets many of our goals. We can see the overall raw figures for each class in Table VII. We can see that, in the OvR case, the numbers are better across the board, with very few exceptions. We can also see how poorly our multiclass classifier seemed to perform on the test set, with some classes showing remarkably few true positives; for example, class 2 has only two true positives. Interestingly, this class also has very few false positives, so the precision and specificity are very high for this class.

If we look at the confusion matrix in Figure 6, the difference is very clear. The multiclass MobileNet demonstrates an unacceptable performance for almost every class, except perhaps for class 3, which has a good number of true positives and few false positives. Class 4, on the other hand, is a significant problem, as it has a perfect score for true positives but a very

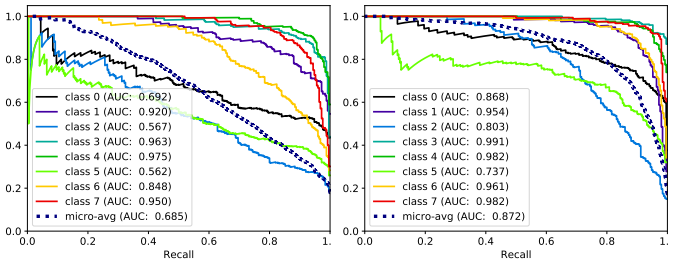
large amount of false positives.



(a) Multiclass confusion matrix (b) Binary confusion matrix

Fig. 6: Confusion matrices for Mobilenet.

On the other hand, the OvR classification is much more acceptable. We still have a slightly high number of false positives for classes 0, 4 and 5. Classes 1 and 2 also have a number of false negatives; however, the results are much more in line with our expectations for a classification problem like this. This effect can also be seen in the PR curves in Figure 7. Here, we see that classes 0 and 2 are the most difficult for our classifier. In summary, we have examined an example in which the OvR style made a great classifier worse, and another in which it made a very bad classifier much better. Judging by the table of average differences, the latter case seems to be slightly more common, and in the cases in which the classifier performed worse, it was usually not by an unacceptable amount.

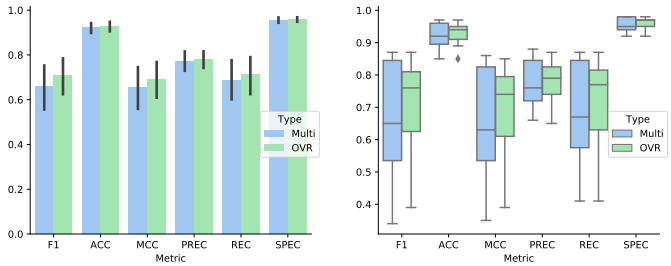


(a) Multiclass PR curves (b) Binary PR curves

Fig. 7: PR curves for Mobilenet.

2) Discussion: It is interesting to examine the raw numbers, but it is difficult to intuit how using an OvR strategy effects the accuracy metrics based on them. In Figure 8, we present two plots that should help us with this. The first, a bar plot with error bars, shows us the small improvements to the average values we saw in the previous sections. We also see that the error bars are quite large for some of these metrics, which makes sense as there is little data.

However, the second plot tells an exciting story. The box plot shows us one of the effects that the action of switching to an OvR strategy has on accuracy. We can see that this changes the distribution of the accuracies significantly. The majority of the observed values fall into a smaller range on the OvR side of the plot, both in the positive and negative senses. This means that the best performers in the multiclass style have become slightly worse, and the worst performers have become slightly better. This seems to reflect our previous observations on the effects. We can see that, specifically, the second quartile (or median) in the box plot has become much higher for many of the metrics.



(a) Bar plot of average metrics on the test set for both OvR and multiclass network styles

(b) Box plot of metrics on the test set for both OvR and multiclass network styles

Fig. 8: Summary plots of the testing metrics.

C. Goals and Recommendations

At the beginning of the experiment section, we discussed some of the requirements in terms of accuracy and speed for the medical scenario. In Table VIII, we have outlined which networks achieved which requirements for the medical scenario. The same effect is visible here as in the previous section, that the best networks from multiclass now fail to achieve some of the same classification goals. However, overall, the number of classification goals not met has reduced from 34 in the multiclass case to 31 in the OvR case. A few of the cases in which we met only the minimum goals have also changed to meet the desired goals in the OvR case, particularly for specificity. Unfortunately, fewer architectures meet our minimum goals for recall in the OvR style, with only one classifier achieving this. In our specific case, the best classifier was a multiclass style network, NASNet Mobile, which met all of our goals, in addition to being extremely fast. We aimed to have high F1 and MCC scores in our results, and, as we can see, the number of architectures that met our goals increased for both of these scores in the OvR style. In terms of speed, the effect of the OvR style becomes apparent, i.e., from almost all networks exceeding the desired FPS goals, only two reach this this goal now, and six do not meet our requirements for real-time processing. The best classifier in the OvR configuration was NASNet Large, which was able to achieve only 6 FPS during testing. MobileNet in the multiclass style is over 100 times faster by comparison, although it is not a particularly good classifier.

Giving direct recommendation on when to use which type of system depend on many paramenters, but based on our results discussed above, we can at least recommend that:

- If your application requires as high classification accuracy as possible (tuning the last persentages), you achieve

	Goals	F1	ACC	MCC	PREC	REC	SPEC	FPS
MULTI	VGG16	Des.	Des.	Des.	Des.	Des.	Des.	Des.
	VGG19	Des.	Des.	Des.	Des.	Des.	Des.	Des.
	Inception v3	N/A	N/A	N/A	Des.	No	Min.	Des.
	DenseNet 121	N/A	N/A	N/A	Des.	No	Min.	Des.
	DenseNet 169	N/A	N/A	N/A	N/A	No	Min.	Des.
	DenseNet 201	N/A	N/A	N/A	N/A	No	Min.	Des.
	Xception	N/A	N/A	N/A	N/A	No	Min.	Des.
	Inception RN v2	N/A	N/A	N/A	N/A	No	Min.	Des.
	Mobilenet	N/A	N/A	N/A	N/A	No	Min.	Des.
	NASNet Large	Des.	Des.	Des.	Des.	No	Des.	Min.
	NASNet Mobile	Des.	Des.	Des.	Des.	Des.	Des.	Des.
OvR	VGG16	Des.	N/A	Des.	Des.	No	Des.	Min.
	VGG19	Des.	Des.	Des.	Des.	No	Des.	Min.
	Inception v3	Des.	N/A	Des.	Des.	No	Des.	Min.
	DenseNet 121	N/A	N/A	N/A	N/A	No	Min.	No
	DenseNet 169	N/A	N/A	N/A	N/A	No	Min.	No
	DenseNet 201	N/A	N/A	N/A	N/A	No	Min.	No
	Xception	N/A	N/A	N/A	Des.	No	Min.	No
	Inception RN v2	N/A	N/A	Des.	Des.	No	Min.	No
	Mobilenet	Des.	N/A	Des.	Des.	No	Des.	Des.
	NASNet Large	Des.	Des.	Des.	Des.	Des.	Des.	No
	NASNet Mobile	Des.	N/A	Des.	Des.	No	Des.	Des.

TABLE VIII: Summary of which requirements were achieved for each network style and architecture. Green cells (Des.) met our desired goals, yellow cells (Min.) met our minimum goal, red cells did not meet any goals, and cells labeled N/A did not meet our desired goal but did not have a minimum goal specified.

better results using an OvR type of system, i.e., muliple binary classification pipelines in parallel.

- If you have any type of system requirements in addition to high accuracy, e.g., real-time feedback, a single multiclass classification system still gives you a high accuracy, but at a lower resource consumption.

ph: Any other recommendations we can give?

V. Conclusion

We have provided a comprehensive comparison of a single multiclass versus OvR classifiers using different deep learning architectures, evaluating both classification accuracy and system resource consumption and speed. Based on our final results, we observe that the performance impact of using the OvR style is rather significant. On average, our classification slowed almost proportionally to the number of classes. However, for the classification experiments, the results showed that the classification accuracy increased on average and that the best networks of both styles achieved almost the same performance, meeting the goals for our medical use case scenario.

References

[1] Endoscopic camera — vimex endoscopy. http://vimex-endoscopy.com/aparatura-endoskopowa/kamera/?lang=en. (Accessed on 05/11/2018).

[2] Mirocam capsule endoscope camera offers a broder field of 170 degrees which enables a more through diagnosis of the small bowel. http://www.reyyanmedical.com/index.php?yazigoster=mirocam-endoscopic-camera&dil=en. (Accessed on 05/11/2018).

[3] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016.

[4] Robbie Allen. Nips accepted papers stats machine learning in practice medium. https://medium.com/machine-learning-in-practice/nips-accepted-papers-stats-26f124843aa0. (Accessed on 05/20/2018).

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

[6] Nvidia Corporation. Geforce gtx 1080 ti graphics cards — nvidia geforce. https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1080-ti/. (Accessed on 05/07/2018).

[7] Juerg Haefliger, Yves Lehareinger, Patrick Blessing, Peter F. Niederer, Daniel Doswald, and Norbert Felber. High-definition digital endoscopy. In *Proceedings of SPIE 3595, Biomedical Diagnostic, Guidance, and Surgical-Assist Systems*, 1999.

[8] Jin-Hyuk Hong and Sung-Bae Cho. A probabilistic multi-class strategy of one-vs.-rest support vector machines for cancer classification. *Neurocomputing*, 71(16-18):3275–3281, 2008.

[9] Michal F. Kaminski, Jaroslaw Regula, Ewa Kraszewska, Marcin Polkowski, Urszula Wojciechowska, Joanna Didkowska, Maria Zwierko, Maciej Rupinski, Marek P. Nowacki, and Eugeniusz Butruk. Quality indicators for colonoscopy and the risk of interval cancer. *New England Journal of Medicine*, 362(19):1795–1803, 2010.

[10] Stephen Marsland. *Machine Learning: An Algorithmic Perspective, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[11] K. Pogorelov, O. Ostroukhova, A. Petlund, P. Halvorsen, T. de Lange, H. N. Espeland, T. Kupka, C. Griwodz, and M. Riegler. Deep learning and handcrafted feature based approaches for automatic detection of angiectasia. In *Proceeding of the IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pages 365–368, March 2018.

[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. Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection. In *Proceedings of the ACM Multimedia Systems Conference (MMSYS)*, pages 164–169, 2017.

[13] 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.

[14] Rees, CJ. et. al. Narrow band imaging optical diagnosis of small colorectal polyps in routine clinical practice: the detect inspect characterise resect and discard 2 (discard 2) study. *Gut*, 66, 2017.

[15] Michael Riegler, Konstantin Pogorelov, Sigrun Losada Eskeland, Peter Thelin Schmidt, Zeno Albisser, Dag Johansen, Carsten Griwodz, Pål Halvorsen, and Thomas De Lange. From annotation to computer-aided diagnosis: Detailed evaluation of a medical multimedia system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 13(3):26:1–26:26, May 2017.

[16] Michael Riegler, Konstantin Pogorelov, Pål Halvorsen, Thomas de Lange, Carsten Griwodz, Peter Thelin Schmidt, Sigrun Losada Eskeland, and Dag Johansen. Eir - efficient computer aided diagnosis framework for gastrointestinal endoscopies. In *Proceedings of the IEEE Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, 2016.

[17] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

[18] van Doorn, SC. et. al. Polyp morphology: an interobserver evaluation for the paris classification among international experts. *Am J Gastroenterol*, 110, 2015.

[19] Jianhua Xu. An extended one-versus-rest support vector machine for multi-label classification. *Neurocomputing*, 74(17):3114–3124, 2011.

[20] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.