A Review of Custom Vision Service for Facilitating an Image Classification

Matea Pejčinović
Asseco SEE
Vukovarska 269D, 10000 Zagreb
matea.pejcinovic@gmail.com

Abstract. *In past few years machine learning has become a buzz word in the industry. There are so many fields that use some algorithm or some approach in order to facilitate the business and improve our everyday lives. One of those fields is image classification that helps us detect important parts of the image using convolutional neural networks. In this paper I will give an overview of the Custom Vision Service built on convolutional neural networks that helps engineers build a model that meets their needs best. This model can then be used for image classification with certain accuracy which will be presented in result.*

Keywords. Machine learning, image classification, Custom Vision Service, convolutional neural network, precision, recall

1 Introduction

In tech world every new release of a new cell phone and similar electronic gadgets comes with interesting innovations that seem to be improved constantly. If we take a closer look, we would get an impression that a lot of those innovations rely on machine learning. There are many fields that take advantage of state-of-the-art algorithms that machine learning provides. But not so many of them are quite present in our daily routines like image classification and recognition.

There has been a massive stride in recent years in medicine, car industry and electronics which is a consequence of a development of solutions that are based on what can be seen. It is not sufficient just to process data as it was in the past, in the beginning of the machine learning era. If we want a computer to know the difference between, for example, a stop traffic sign and a speed sign (which is rather important for self-driving cars), we need to simulate the process of logical reasoning that is common to us humans (Kriesel, 2018). For such purposes, we do need some form of intelligence and this is where deep learning takes its place.

Deep learning uses neural networks; convolutional neural networks to be more precise, in order to yield adequate results in a field of image classification. Like with any other approach, a lot of training is required to bring about accurate precisions that neural networks will have to use to learn over time. They enhance their performance by learning on a training set without which they would not know how to classify objects presented on images.

When we think of images, we usually label objects and recognize specific patterns. Convolutional neural network will break images into numbers and proceed with image processing. It compares these numbers' parts, called features, and tries to find a match roughly at the same positions in the images that are being compared to. At start, a convolutional neural network cannot know where those feature that it should compare are. So, it makes logical to try to find them across the whole image. When calculating a match, a convolution is applied (Goodfellow et al., 2016). This means that a certain math is behind it and that it tries to multiply the numbers in the feature with the number in the corresponding part of the image. If they match, we should get the result 1, and if they don't match, we should get -1. Using this math, we could find out the position of the object in the image.

Aim of this paper is to present the API that can facilitate the process of building a model for image classification. Therefore, an engineer can pay a lot of attention to the training set and tag images properly so that the model can provide more accurate predictions. This paper is organized as follows. Firstly, the convolutional neural network is presented as most important approach in the image classification. Then, a short literature overview will be listed. Next chapter deals with the most important features of the custom vision API, the state-of-the-art solution based on convolutional neural network. Afterwards, statistics with dataset explanation and result presentation will be shown along with the analysis of the results. A separate subchapter will compare the performance of the model built with Custom Vision Service with the model based on SVM. Finally, in the last section, certain conclusions are displayed.

2 Convolutional Neural Network

It is wide known that simple recognition tasks can be solved with satisfactory level of accuracy when a model is trained on small datasets (in order of tens of thousands of images). But when there is certain need to perform classification and recognition task on objects that are shown in a realistic environment, such a small dataset does not provide results we could consider sufficiently good (Pinto et al., 2008). Therefore, one could come to conclusion that a vast dataset would definitely work when it comes to object recognition on the images. But this is not the case. Actually, not only there is a necessity of a great dataset, but also a model that would process that dataset would have to have the ability to learn from the features present on the image and take an advantage of the prior knowledge in recognition of lost data that are not available for the provided dataset's items.

Such a model can be built using convolutional neural networks, a type of a neural network with a huge learning capacity. They represent an important class of algorithms which have been shown to be state-of-the-art on large object recognition, image classification and many other tasks related to images and natural language processing. They were not so accepted in the past years due to their expensiveness in application to certain images. Nevertheless, the improvement of technology and datasets have led to popularisation of this approach without fear of an overfitting. These neural networks can be parametrized for determination of their depth and breadth. It is possible to improve their performance by removing certain layers and using number of features that are rather unusual, but which will possibly reduce the training time. Their main advantage is that they learn using quite general features and they are nowadays widely used due to their generalisation ability on unseen data. Due to this fact, significant attention has to be paid to preparing a dataset which needs to be carefully selected so that the model trained on it could be easily used for all kinds of image classification tasks. Images need to differ in light, a camera angle, size, object position etc. This is how a convolutional neural network could learn and, later on, apply this knowledge to the images in the test dataset. Images will be converted to numbers and each pixel will participate in convolution (Goodfellow et al., 2016). This math is actually the base for the determination of the class for each image. Since the aim of this paper is not to present convolutional neural network but the service built on them, there will be no detailed overview of this algorithm but just short description of related work in the next chapter.

3 Related work

Many scientists have already worked in the field of machine learning presented in this paper. Image classification is quite attractive because it can be applied to many tasks. There have been some research studies that deal with face recognition based on convolutional neural networks. Most of them rely on geometrical features of face, such as a position of mouth, shape of chin etc. (Brunelli & Poggio, 1993). In (Lawrence et al., 1997) authors have decided to use a combination of a local image sample representation and a convolutional neural network for face recognition. Their model resulted in 10.5 % error.

Excellent results were also achieved in other paper (Krizhevsky et al., 2012) where authors built a deep convolutional neural network that had outstanding results, but they also noticed that if they remove just one layer, the performance is significantly getting worse. Like with many other papers, they do believe that they could train their network longer to improve results. Convolutional neural networks were also used for hyperspectral image classification (Hu et al., 2015). It was shown that this approach offers excellent performance even though there was only one convolutional layer and one fully connected layer. A new method to train convolutional neural network from scratch using low-rank filters is proposed in another paper (Ioannou et al., 2015). Authors have shown that with fewer, more relevant parameters it is possible to prevent overfitting, increase generalization and increase accuracy.

Not only are the convolutional neural networks interesting in image classification, but also in sentence classification. There are many papers that deal with a series of experiments with these neural networks, like (Collobert & Weston, 2008) and (Kim, 2014). The author of the last-mentioned paper used them on already prepared word vectors from *word2vec* and achieved some great results with just one layer of convolution.

4 Custom Vision Service

As it was stated previously, for an image classification task it is rather important to build a proper model. This means that our model should ensure a satisfactory accuracy of prediction. Many of us engineers neither have enough experience in machine learning nor enough time to build a model for every possible case. There are many available solutions but they are usually tied to a specific case and they do not provide good results for different tasks in a computer vision field. Therefore, an engineer has to build his/her own model which would not probably return presumed results due to several adjustments missing.

This problem can be overcomed these days by using cognitive services hosted in Azure ("What is Azure Custom Vision?", 2019). This solution is basically a set of several APIs that support developers and engineers in integrating artificial intelligence features within their application. Among those APIs is a Custom Vision API as well.

Microsoft Azure's Custom Vision API ("Cognitive Services Custom Vision", 2019) is a cloud-based API that helps engineers building a model that would lead to improvement of custom image classifiers. This means that we are entitled to provide a training and a testing dataset which needs to be properly labelled because these tags actually represent classes. We are the ones that are expected to determine those labels at the time of the image insertion to the database. At that point, a convolutional neural network will apply those labels and train the model.

Custom Vision Service is available as a set of native SDKs as well as through web-based interface ("Build a Custom Vision Service classifier", 2019). Before we are able to use this service for image classification, a model needs to be built. It is possible to build a model for image classification or for an object detection task. Labels could be applied in a form of a single tag or multiple tags per image. Classification task could refer to several domains that are optimized according to the specific type of images. These domains are generic, food, landmarks, retail, adult and compact domain.

For a shopping catalogue or a shopping website, it is strongly advisable to train a model in a retail domain. This model has been already optimized for classification between all types of clothes. Adult domain should be applied to models that should block inappropriate content that, for example, children should not see. Food domain differentiate dishes, fruits and vegetables while compact domains can come quite handy since they are optimized for constraints of real time classification on mobile devices. As it is perfectly clear from its name, landmarks domain help building classification model for both natural and artificial recognizable landmarks with high accuracy rate with even slight obstructions by people in front of those landmarks. Unlike the others, generic domain is optimized for a broad range of classification tasks.

Azure portal allows inspection of keys, access to the API reference and to the Custom Vision Portal etc. It is possible to track the logs and take a look at the real time API usage. Additional useful pieces of information are available once a project is created.

After the creation of the project, we are able to upload the images and tag them. This API is highly optimized to quickly recognize major differences between images. But for such advantage of this API, at least 50 images per tag should be in the dataset. Every number less than 50 will not provide results with satisfactory level of precision. Like it is case with other machine learning implementations, this one also has some issues with images that slightly differ between each other. But, the quality of the classifier can be improved. It is highly dependent on the quality of the images, their number in the dataset as well as the variety of the labelled data. The dataset has to be unbiased, which means balance should be established between data so they can be considered the reliable representations of data expected in the test set. In order to accomplish such results, the model built on convolutional neural network from Custom Vision API should be trained several times. As it was stated before, this dataset is highly dependent on the training dataset. Since the engineers do not need to build model on their own, it is solely their responsibility to provide reliable data for training. After first time training, it is recommended to adjust the training set in order to achieve the balance. There should be enough data for each label so that convolutional neural network can take into account all the differences between labels easily. This is usually achieved by added more images, especially images to the label that lacks data. After this, another round of training should be performed and after it, another adjustment of the dataset could be done. This refers to the images that differ from those already in the dataset in form of the size of the objects, a background and the lightning. After all these dataset improvements, one more round of training should be performed and the model is ready for testing.

The submission of the images for testing can be done programmatically or manually. If we are interested in submitting data programmatically to the Prediction API, an iteration for prediction should be published and this is how the model, ready for testing, is accessible to the Prediction API. At this moment, we can submit images for testing purposes to the REST API using Prediction URL and Prediction-Key.

Results are returned in a form of a JSON document with the identifiers of the project, iterations and predictions.

The major advantage of the Custom Vision Service is that the classifier can be exported to run offline. Therefore, it can be embedded in any application and run locally on a device for real-time classification. It supports following exports (only for compact domains):

- Tensorflow for Android ("TensorFlow", 2019)
- CoreML for iOS11 ("Core ML", 2019)
- ONNX for Windows ML ("ONNX", 2019)
- a Windows or Linux container

Another advantage of this service is also the fact that it is not language dependent. The published API can be used in any programming language when an iteration is published and keys are available to use.

In the next chapter a dataset used for the purpose of reviewing this API and its features will be presented. An evaluation of results and comparison with SVM is also shown along with some further steps.

5 Model design and evaluation

Since image classification is of great importance to the car industry in the field of the self-driving cars, it seemed as an appropriate decision to make an image classifier for the traffic signs. It is rather important to know whether a self-driving car should slow down, if

there are some construction works on the road, if there is a stop sign in front etc.

This section explains the research of the Custom Vision Service features and possibilities in the field of traffic signs' classification. It will contain a description of the dataset (training and testing data) and several training rounds. Custom Vision Service from Microsoft Azure platform is used for development of the classification model. At last, the model will be evaluated and the results will be displayed. A comparison of results with SVM will be presented.

5.1 Dataset description

Custom Vision API allows easy training of a classifier using your own data. The model has been actually built upon deep neural networks that can have several layers. Since the topic of this paper regards a traffic signs' recognition, there were no previously optimised models already built, unlike some other image classification's topics. Therefore, the success rates are highly dependent on the dataset that is used. A dataset needs to have a plethora of images highly variant upon lightning, brightness, camera angle etc. It is quite clear that for such discrepancy between images the numerous images should be applied in training phases. For such purpose, the dataset used for training and testing was designed by merging four different image sets. The first one was a collection of images from an online traffic school. These images present traffic signs currently in use in the Republic of Croatia. The other ones were:

- the Kaggle's datasets for traffic sign recognition ("Kaggle traffic sign recognition dataset", 2016)
- the Belgium traffic signs dataset ("KUL Belgium Traffic Sign dataset", 2010)
- the German traffic signs dataset ("The German Traffic Sign Detection Benchmark", 2015)

Main reason for merging those datasets was to have enough data with highly distinctive features that could allow the model notice the discrepancy between the traffic signs and successful classification. As it was previously stated, the model's success classification rates depend on the variety of images in a dataset and their tagging. The more divergent images for the same traffic signs are (in a form of the lightning, camera angle, brightness etc.), the easier it would be to find specific features that could separate that sign from the others. A screenshot representing a simple part of the dataset with images representing the same traffic sign in different positions and some other features can be seen on Fig. 1:

Figure 1. A part of the dataset showing variant images for the same traffic signs

Since it is rather important to also learn on the images that present real world scenario, with the landscape and the traffic, the road and the pedestrians, some of such images were also added to the dataset. These images are publicly available and downloadable from Google. A merged dataset was initially a set of the 360 images. Some of the datasets were already divided in a training and a testing set, while the others, i.e. the Croatian traffic signs dataset, were split in a ratio of 3:1. This ratio has been chosen in order to have enough data for training, as well for testing purposes.

Knowing that it would be needed to go through several rounds of training since the process of building a classifier is an iterative process, a whole dataset for training was not used for the first round. It only contained images presenting different traffic signs that differ in size and a background. Data distribution for the first round is presented on a figure below:

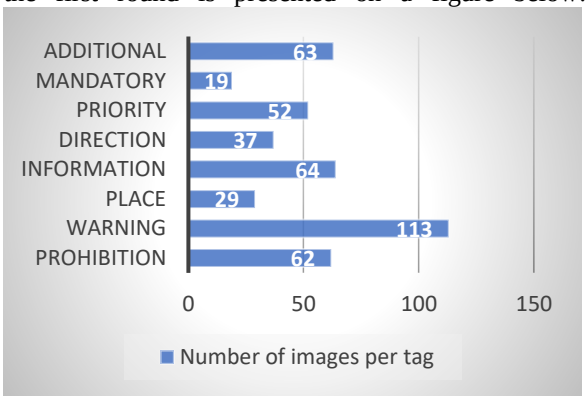

Figure 2. Data distribution for the first round of training

Testing phase after the first round of training has shown that some additional images were needed in the dataset. For the first round of training, the dataset was not properly balanced. This was because there were not at least 50 images per tag present, as it can be seen on a Fig. 2. The place, mandatory and direction tags were lacking training data which could enhance the performance of the built model. Nevertheless, the first iteration resulted with the precision rate of 82.2% and the recall rate of 81.9%.

The model has not been previously optimised, but there are some recommendations that should be followed in order to build a model with better measures (k-fold cross validation) ("How to improve your classifier?", 2019). This specifically refers to having at

least 50 images per tag, which allows neural network that is responsible for training the model, to recognize specific features of each tag and to classify more accurately testing images. This limitation is logical, because more images you have, easier it would be to find out what differentiate warning signs from direction signs. A dataset with fewer images per label is under higher risk of overfitting which could lead to a model that may struggle with real-world data.

According to the previously mentioned recommendations, an adjustment for the next training round was made. Additional images were added, but not all kind of images. The dataset images should differ in terms of camera angle, lighting, background, size and individual/grouped items. Using this, a balanced dataset was built in order to get an unbiased model. So, after adding new images, the new dataset was in balance and the classification model is ready to use. Almost all labels have approximately the same quantity of training data. A distribution of labels was somewhat even, just warning signs were exceeding in number as it is shown on a Fig. 3. Having the data distribution like this, the model will not be more accurate in predicting one label than another. Since the recommendation of having at least a 1:2 ratio between the label with the fewest images and the label with most images is accomplished, the performance of the model should be satisfactory. For the second round of training, a hundred and eight new images were tagged. They present traffic signs in different lightning, brightness and other, already stated, features.

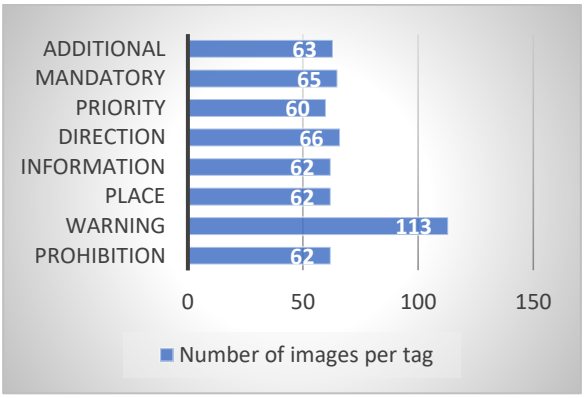

Figure 3. Distribution of the training dataset after second round

For the third round of training, another set of the 350 images were applied for training representing real world scenarios with a lot of noise on the image. Third and last round was undertaken in order to improve performance of the model and taking into account the measures of accuracy after second round of training.

There is no recommended number of training rounds. When a model achieves an acceptable performance, there is no need for further training. But if there is a certain necessity for improving results, next step would be adding new images for each tag, or specific tag if it lacks data, and proceeding with training.

A final dataset has 818 images, some of which can been seen on a figure below:

Figure 4. A part of the training dataset

The Custom Vision API allows a single tag, as well as multiple tags per each image in the dataset. Since it can be foreseen that several traffic signs could appear on a single image (it is completely valid to see a combination of several traffic signs together), a multiple tags per image were labelled. It is possible to upload images programmatically, as well through very easy-to-use GUI by dragging and dropping images in the dataset.

There are overall eight tags that were used. The sign can be classified as additional, direction, information, mandatory, place, priority, prohibition or warning sign.

Additional label refers to information tables that usually describe some information regarding the length of some prohibition and expected weather conditions. Direction label is applied to traffic signs that show direction to some place or city. Information traffic sign is usually displayed with blue colour and can appear in circular or square form. They do not express any kind of warning or obligation but information a driver should be aware of. Mandatory signs define the obligation for a driver, e.g. an obligation to drive in a certain traffic lane. Place tag is applied to a city or a place name. Priority refers to a stop sign, a main road and the side roads. Prohibition label express, among everything else, the allowed speed while warning tag is applied to data images like road signs and traffic lights.

5.2 Model evaluation

There are several methods for a model evaluation but, in this case, a standard approach was used. This is a k-fold cross validation which estimates how accurately will a classifier perform in the real case scenarios. For each iteration and model training, there is a display for a precision and a recall rate. Precision means the percentage of the results that were relevant (Bonnin, 2017). Recall can be defined as the ability of the model to find all the relevant cases within the dataset (Bonnin, 2017).

Considering the topic of this paper, three adequate metrics were chosen to evaluate the model. Precision and recall have already been mentioned. Third one is F1-score. Precision and recall are identified as relevant measures since it is necessary to find to balance

between them. F1-score is a function of a precision and a recall and is needed exactly for the purpose of finding a balance between those two metrics. Since accuracy as a measure can be largely contributed by a vast number of true negatives, this measure just does not seem appropriate when there is need to focus on false negatives and false positives. Depending on the topic of the paper, it is sometimes a lot worse for a model to detect something as a false negative, rather than a false positive. In this field of machine learning, three before mentioned metrics are better to use than the accuracy of the model since they provide better ways for the model evaluation.

After three rounds of training, the precision rate and a recall rate were quite satisfactory. The model built upon the training dataset can be considered accurate and unbiased. The graph below presents the analysis of precision rate, recall rate and F1-score in all three iterations.

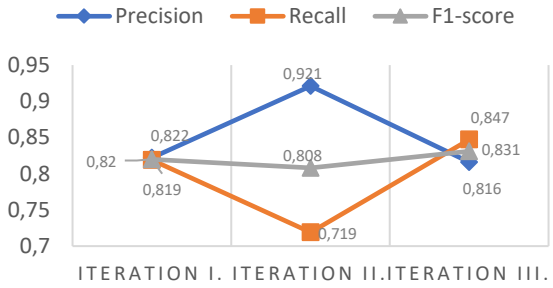

Figure 5. Statistics for all three rounds of training

The figure above presents the relation between these three measures. Clearly, the peak of the F1-score, as a function of precision and recall, is in the third iteration. The graph shows that in that iteration, the best balance between those two measures is accomplished. The other lines show oscillation of precision and recall. They are closer to each other only in the last iteration. Shown values can be considered satisfactory and a model should be unbiased with provided dataset.

Even with a smaller number of images in a dataset, a model built with Custom Vision Service has very good metrics. The precision and a recall actually increase as the amount of the images in training dataset is enlarged. But, as it was previously stated, the model works relatively well even with less images in a training dataset. This fact can be noticed from the Fig. 5 which shows the metrics values in all three iterations. The first iteration has the smallest dataset while in the third iteration the model is being trained on significantly larger set of images which has eventually led to the model with better performance for all tags.

For further evaluation, the impact of the probability threshold is analysed. If there is a test image that needs to be submitted for a prediction and a low probability threshold is set, then the model did find a lot of data that is to be considered a true positive, but there are also false positives detected. High probability threshold returns results at the expense of recall. The graphical analysis of the impact of the probability threshold rate to the recall in the first and the second iteration can be seen on Fig. 6.

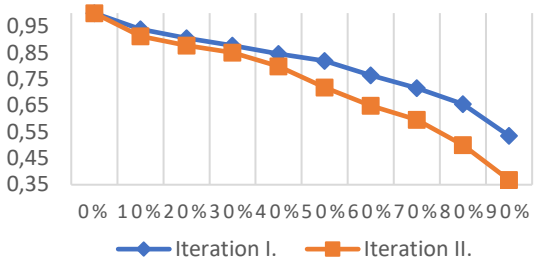

Figure 6. Impact of a probability threshold (x-axis) to a recall rate (y-axis)

This graph shows that the recall is getting decreased in the second iteration. For each value of a probability threshold that is displayed on an x-axis, recall rate is bigger in the first iteration. In both rounds, the recall is decreasing with the probability threshold rate increasing, but in the first round, recall's decrease rate is slower than in the second round of training. While recall gets lower, precision rate is getting bigger with each increase of a probability threshold, which is shown on a Fig. 7.

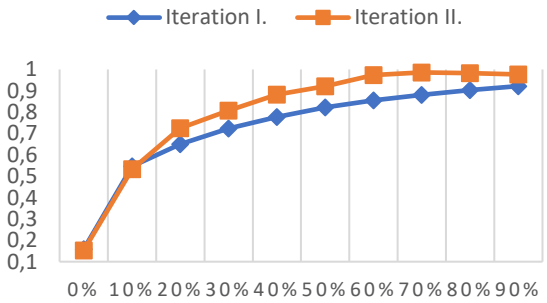

Figure 7. Impact of a probability threshold (x-axis) to a precision rate (y-rate)

5.2.1 First round of training

As it was previously described in a subchapter 5.1, the dataset used for the first round of training was not balanced at all. The model built upon that dataset was rather biased and not very accurate. The distribution of data did not followed recommendations about having at least 50 images per tag. Putting aside the fact that the data were well labelled, the model did not have enough data to be trained well for all eight tags. This is the reason why, using the model gained from the first round of training, some images are precisely accurately classified (belonging to those tags with enough training data), and the others are not. The model's measures for each tag for the first round of training can be seen in Table 1:

Table 1. Model's metrics with probability threshold 50 %

Tag	Precision rate	Recall rate	F1-score
Prohibition	0.934	0.934	0.934
Warning	0.890	0.893	0.891
Place	0.882	0.972	0.925
Information	0.826	0.720	0.769
Direction	0.813	0.845	0.829
Priority	0.787	0.788	0.787
Mandatory	0.722	0.933	0.814
Additional	0.650	0.569	0.607

The impact of a probability threshold can be huge, depending on which measure is considered more relevant for the model. Below are three figures that are graphical presentations of the metrics for some of the tags where it can be seen how metrics differ upon variation of a probability threshold.

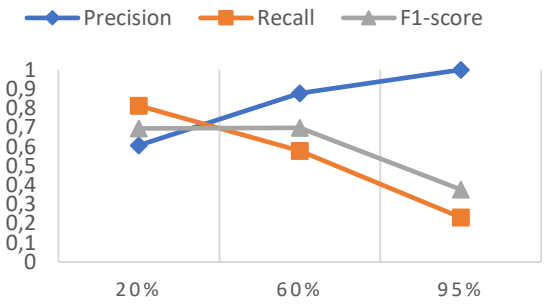

Figure 8. Information tag's metrics with different probability threshold values (x-axis)

Fig. 8 presents three line graphs where an x-axis holds the values for the chosen values of probability threshold, while a y-axis is for displaying the rate for each measure. Blue coloured graph presents precision, the orange one is a recall and the gray one is an F1-score. These graphical visuals illustrate the bound between a precision and a recall. A slight variation of a probability threshold can sometimes lead to the significate changes in the measures' rates. It is clear that, when a probability threshold is being enlarged, a precision rate is also rising, while a recall is lowering down, and vice versa. Fig. 9 and Fig. 10 present the same relation between a precision rate, a recall rate and an F1-score, but for a priority and a place traffic sign's tag.

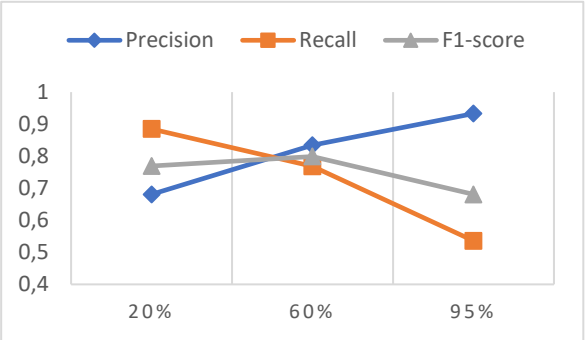

Figure 9. Priority tag's metrics with different probability threshold values

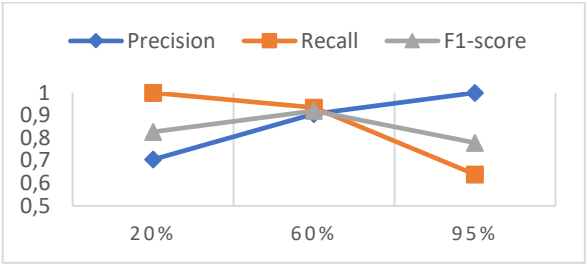

Figure 10. Place tag's metrics with different probability threshold values

Overall values of model's metrics for this iteration with probability threshold of 50% are as follows:

$$P = 0.822, R = 0.819, F1 = 0.8205$$

5.2.2 Second round of training

The second round of training has been shown as must, since there have been three labels that lacked data for training the model in the first iteration. As it was previously stated, such a dataset would lead to a biased model as a consequence of an unbalanced dataset. Therefore, additional images were added, mostly in those tags that were missing data, but still using the same sources stated in a subchapter 5.1. The images were mostly added to the labels that were lacking data in the first iteration. These images were chosen with certain data refining taking into account that they vary by lightning, background, size, camera angle, visual style etc.

Adding such images, the model has had at least 50 images per tag which allowed building an unbiased model upon such balanced data. Table 2 displays the measures for this round of training:

Table 2. Metrics for each tag in second iteration

Tag	Precision rate	Recall rate	F1-score
Prohibition	0.923	0.923	0.923
Warning	1.000	0.957	0.978
Place	1.000	0.846	0.917
Information	0.833	0.385	0.527
Direction	0.692	0.692	0.692
Priority	1.000	0.583	0.737
Mandatory	1.000	0.846	0.917
Additional	0.833	0.357	0.499

It can be seen that, for this iteration, the precision rate has been enlarged since more images were added to each tag.

In this iteration, following rates have been achieved (with standard probability threshold value of 50%):

$$P = 0.921, R = 0.719, F1 = 0.808$$

Precision rate has risen with recall rate being decreased. Even for this iteration, it is very interesting to show the impact of a probability threshold rate on

the overall precision and recall values which is shown on the line graphs on Fig. 11 and in Table 3. It seems the most balanced ratio between precision and a recall is gained with the probability threshold between 30% and 40%. If it is more important to decrease the number of false positives, then this threshold should be even bigger in order to increase precision.

Table 3. Probability threshold's rate impact

Probability threshold	Precision rate	Recall rate	F1-score
0 %	0.153	1.000	0.265
10 %	0.533	0.912	0.673
20 %	0.725	0.877	0.794
30 %	0.808	0.851	0.829
40 %	0.883	0.798	0.838
50 %	0.921	0.719	0.808
60 %	0.974	0.649	0.779
70 %	0.986	0.596	0.743
80 %	0.983	0.500	0.663
90 %	0.977	0.368	0.535

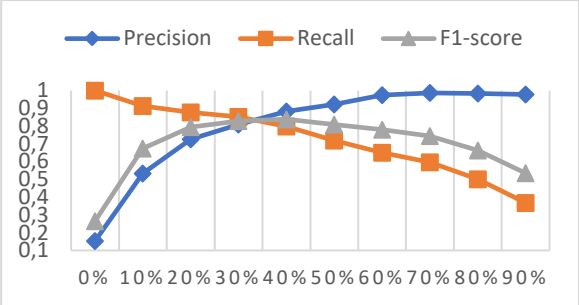

Figure 11. Model's metrics with different probability threshold values

5.2.3 Third round of training

For the third round of training, a lot of images with different background, lightning, brightness and camera angle were added to each label. This addition resulted with the following distribution of the images in the dataset:

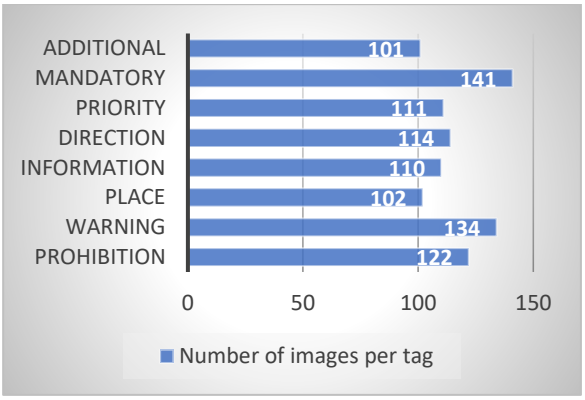

Figure 12. Data distribution in the third iteration

All labels have more than hundred images and the images are almost equally distributed. Adding a few images to the dataset has led to an increase of a recall, while a precision has been decreased.

Table 4. Model's metrics by tags in third iteration

Tag	Precision rate	Recall rate	F1-score
Prohibition	0.759	0.880	0.815
Warning	0.897	0.963	0.929
Place	0.800	0.952	0.869
Information	0.680	0.773	0.724
Direction	0.727	0.696	0.711
Priority	0.875	0.955	0.913
Mandatory	0.923	0.857	0.889
Additional	0.875	0.667	0.757

Taking a look at the above presented table, it can be seen that the metrics are being more balanced (the measures for the labels are more even than in previous iterations). The overall metrics for this iteration is shown below:

$$P = 0.816, R = 0.847, F1 = 0.831$$

It is clear that the F1-score has been enlarged in this iteration which definitely means that the model is, with this iteration, having better ratio between precision and recall. If necessary, further iterations can be made and more trainings can be performed but for the purpose of this paper, these metrics values can be considered satisfactory.

Table 5. Model's measures depending on a probability threshold rate

Probability threshold	Precision rate	Recall rate	F1-score
0 %	0.141	1.000	0.247
10 %	0.535	0.942	0.682
20 %	0.658	0.905	0.762
30 %	0.726	0.884	0.797
40 %	0.759	0.868	0.810
50 %	0.816	0.847	0.831
60 %	0.847	0.820	0.833
70 %	0.877	0.757	0.813
80 %	0.903	0.693	0.784
90 %	0.918	0.593	0.721

As it was presented for the other iterations, the impact of the probability threshold's rate on the model's is displayed for this iteration as well. In Table 5 it can be seen the F1- score is more even than in the other two iterations. This fact could be considered a consequence of the additional images in each label which eventually has led to more realistic values of precision and a recall rate.

5.3 Evaluation on test set

After the training has been performed, it is required to test this model on a test dataset. This means that it is necessary to test the model on images a model had not seen before but which are highly expected to be classified correctly upon prior knowledge. Convolutional neural network should find the features for the test images and compare them with the knowledge gained on the training dataset. The Custom Vision Service based model performs an evaluation on single or multiple images. It needs to be submitted to the model programmatically via API URL or using GUI ("Test and retrain a model with Custom Vision Service", 2019). No matter how it is submitted, the result is presented in a form of percentage per tag. Model gives the highest rate to the tag that the model should be classified to. If there are several traffic signs on the image, the model will give a substantial rate to each of the traffic signs' tags that were recognized and, therefore, classified.

Using the first iteration of the classifier it was shown that the traffic signs related to direction label were not classified with great accuracy. Images that were showing a prohibition were classified with high precision level. Such an outcome was not surprise because the dataset was not balanced.

Since the model was biased, in the second round of model building, more images were added to the labels that were lacking data, as it was described in previous chapter. After that, a testing prediction task was started and it was shown that the images, where a sign is in front, are usually classified with the probability rate of at least 84 %.

But, when testing the model on images with a lot of additional content, not only traffic signs (e.g. cars, landscape etc.), the probability rate is decreased for maximum half of the previous value. Still, the signs were always correctly classified. This means that the highest probability rate was always assigned to the proper tag.

Some of the testing data are presented below:

Figure 13. Part of the test dataset

Finally, these images were tested with the model from the third iteration. This iteration gives very good results on a test dataset. As it was already stated, the model from this iteration has following measures:

$$P = 0.816,\ R = 0.847,\ F1 = 0.831$$

Fig. 14 shows an image from a test dataset with several traffic signs. The model has successfully classified three groups of traffic signs with rates listed in Table 6.

Figure 14. Submitted test image to the model from the third iteration

Table 6. Model's prediction for test image

Tag	Probability
Warning	97.9 %
Additional	84.7 %
Priority	36.7 %
Prohibition	15.8 %
Direction	4.5 %
Mandatory	2.0 %
Information	0.7 %
Place	0 %

Considering the images from the training dataset, this result is rather satisfactory. The image had three groups of traffic signs and they are all been given quite good probability rate. This result presents real-world scenario for which a model showed acceptable prediction rates given the fact that it should have been tested further to improve its performance for such cases. The model, however, shows great probability rate for images where signs do not cover each other and where is not a lot of noise. The signs are correctly classified with significant percentage given to the most probable label, like it is shown in Table 7 for a Fig. 15:

Figure 15. A direction traffic signs submitted to model

On an above presented figure, an image with several direction traffic signs was submitted for an image classification task. The results, presented in Table 7, are showing that the model correctly classified

the image to the direction label with the probability rate of 99.8 % while the other labels have been given less than 2% probability rate, except prohibition and mandatory tag with 0 %.

Table 7. Model's prediction for a test image

Tag	Probability
Direction	99.8 %
Information	1.8 %
Additional	1.6 %
Warning	0.7 %
Priority	0.6 %
Place	0.1 %
Prohibition	0 %
Mandatory	0 %

It seems that the model, after three iterations, classifies the traffic signs with good precision. It can be improved, though, for some images with a lot of noise since some tags can be recognized with higher probability rate when there are multiple signs on the image. Therefore, the model performs well for the multiclass tasks, but needs further training for better performance for multilabel test cases.

However, real evaluation of the model can be accomplished in comparison with the models built with other machine learning techniques. Next chapter will present a comparison of performance of this model built on convolutional neural network with the performance of the model built on Support Vector Machine (SVM).

5.4 Model evaluation in comparison with SVM

Support vector machine (SVM) is a machine learning algorithm commonly used for both classification and regression tasks ("Support Vector Machines", 2019). It is a form of a supervised learning that takes a set of training data with examples that consist of items and assigned classification to a specific category and builds a classifier model (Alpaydin, 2009). It is inherently two-class classifiers but an SVM can also solve multiclass problems (Mathur & Foody, 2008). SVM was chosen for comparison of performances with the model based on the convolutional neural network in order to find out how well one of the most common algorithms classifies several classes of traffic signs. It is also often used for classification of images for which SVM achieves high accuracy after several rounds of training. For the purpose of performance comparison, images from the first iteration were used in order to build a model and get the metrics.

Model was built using Python 3 and its widely used libraries from scikit-learn set of tools**.** As it was already stated, the dataset for training was the same as it was used for the first iteration of training described in a chapter 5.2. Those images, 360 of them, were resized to the same dimension (300px X 150px) and then the model has been fitted. The one-against-one approach was applied. This means that 28 classifiers were constructed for 8 labels and each classifier trained data from two classes. The number of classifiers is calculated using the following formula:

$$n\_classifiers = \frac{n\_classes * (n\_classes - 1)}{2} \quad (1)$$

In this case, there are eight classes 28 classifiers. The fitting was processed on the training set after which the prediction phase was ready to start. In this phase, the classifier should have classified the images from the test dataset. They were resized to the same size as the images from the training dataset and submitted for the test. It seemed that the very small images and blurred ones were not correctly classified because in the training dataset from the first iteration, there were few images that were blurred or very small. Therefore, in the training dataset for SVM should be added additional images with different types of distinction between images, e.g. lightning, brightness, size etc. Nevertheless, for comparison purpose, the applied dataset had been sufficient. Table 8 shows the metrics per each tag for a model built with SVM.

Table 8. Prediction with SVM

Tag	Precision rate	Recall rate	F1-score
Prohibition	1.00	0.55	0.71
Warning	1.00	0.57	0.73
Place	0.50	1.00	0.67
Information	0.79	0.78	0.78
Direction	0.72	0.93	0.81
Priority	0.60	0.60	0.60
Mandatory	0.50	0.86	0.63
Additional	0.43	1.00	0.60

These probability rates are relatively good considering the number of images in the training dataset and their variety. Additional images should be placed to the training dataset to achieve better metrics.

The same dataset was applied in the first iteration of the training process for the model built on convolutional neural network and the measures were shown in a chapter 5.2.1. The graphs below compare the performance of this model and the one built with SVM. The recall and precision rates for Custom Vision are presented in a Table 1 while the Table 8 contains values for SVM which are used for the comparison presented on Fig. 16 and Fig. 17.

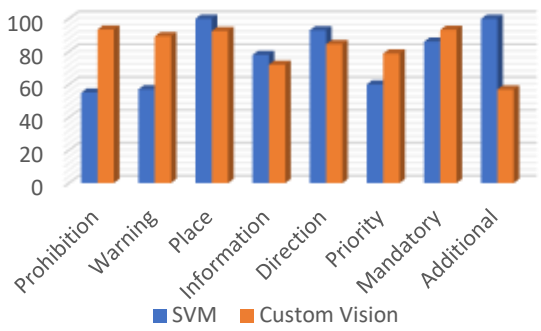

Figure 16. Comparison of recall between SVM and Custom Vision

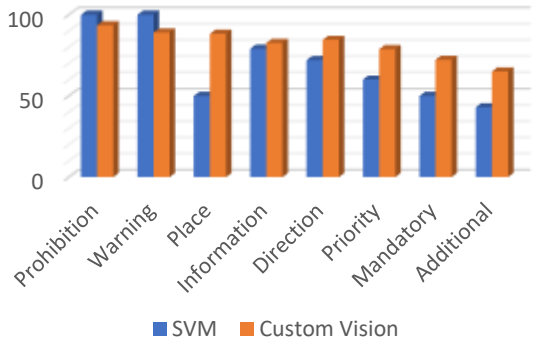

Figure 17. Comparison of precision between SVM and Custom Vision

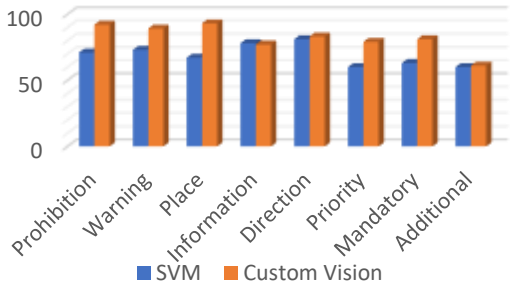

Figure 18. F1-score comparison between SVM and Custom Vision

Histograms on Fig. 16 and Fig. 17 show the values for prediction and recall rates per each tag. Clearly, for some tags, Custom Vision built model exceeds the SVM built model while for the other tags their performance is quite close. The discrepancy between these two approaches can be better distinguished on a Fig 18. It displays the F1-score for labels for SVM (shown in Table 8) and Custom Vision (Table 1). Since it is a function for finding a balance between precision and recall, the evaluation of performance of these two models can be well observed on this graph. SVM is slightly behind Custom Vision (the first iteration) for the same training dataset while the Custom Vision model has more even metrics per tags. SVM and Custom Vision models are most close for information, additional and direction tags since the images for them didn't have much noise. For the others, Custom Vision built model shows better performance than the SVM. For improvement of its probability rate, SVM model should be further trained.

6 Conclusion

The aim of this paper is to present an overview of the new framework, Custom Vision Service, available on Microsoft Azure. It helps engineers a lot in development of image classifiers when it comes to certain tasks for which we do not have an already built classification model.

In the first chapter there was a simple introduction where an importance of the image classification, especially in some fields in industry, was elaborated. Image classification has an important role in our mobile phones' features, in classification of numerous entities, in medicine, but also in the self-driven car industry. Due to the last example, in this paper a traffic sign classification problem was introduced. It is also shown how to build a model using a dataset that needs to be thoroughly balanced. Images that differ in size, light, a camera angle, but also blurring are having great impact on the model and its precision and recall.

It was described how the Custom Vision Service is based on convolutional neural networks, a state-of-the art algorithm with huge learning capacity. Since the building of a convolutional neural network is a tedious task that requires quite some time that engineers most often do not have, a Custom Vision Service provides this part for building a classifier. But an engineer is responsible for the training. It is him who needs to prepare data and label it properly; an engineer needs to provide adequate number of images in the dataset and split it in a proper ratio between training and testing data.

This approach was shown to yield interesting results and precision rate that was quite satisfactory. However, some limitations were shown, e.g. the model should be trained on more real-world scenarios where there could be a lot of elements in front of the sign but the model would still manage to classify the sign with better probability rate. Nevertheless, the results given so far were also very good. They need to be improved so that this model could become a part of something bigger that could make an impact on the industry. Comparison with the SVM showed good performance of Custom Vision model which confirms that this approach in machine learning could be well accepted and further developed.

References

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, *8*(1), 98-113.

Ioannou, Y., Robertson, D., Shotton, J., Cipolla, R., & Criminisi, A. (2015). Training cnns with low-rank filters for efficient image classification. *arXiv preprint arXiv:1511.06744*.

Hu, W., Huang, Y., Wei, L., Zhang, F., & Li, H. (2015). Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors*, *2015*.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160-167). ACM.

Cognitive Services Custom Vision. (2019). Retrieved from https://azure.microsoft.com/en-us/services/cognitive-services/custom-vision-service/

What is Azure Custom Vision? (2019). Retrieved from https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/home

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press

Build a Custom Vision Service classifier. (2019). Retrieved from https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/getting-started-build-a-classifier

Test and retrain a model with Custom Vision Service. (2019). Retrieved from https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/test-your-model

How to improve your classifier? (2019). Retrieved from https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/getting-started-improving-your-classifier

Kaggle traffic sign recognition dataset. (2016). Retrieved from https://www.kaggle.com/c/traffic-sign-recognition

KUL Belgium Traffic Sign dataset (2010). Retrieved from https://btsd.ethz.ch/shareddata/

The German Traffic Sign Detection Benchmark. (2015). Retrieved from http://benchmark.ini.rub.de/?section=gtsdb&subsection=dataset

TensorFlow. (2019). Retrieved from https://www.tensorflow.org

Core ML. (2019). Retrieved from https://developer.apple.com/documentation/coreml

ONNX. (2019). Retrieved from https://onnx.ai/

Brunelli, R., & Poggio, T. (1993). Face recognition: Features versus templates. *IEEE transactions on pattern analysis and machine intelligence*, *15*(10), 1042-1052.

Pinto, N., Cox, D. D., & DiCarlo, J. J. (2008). Why is real-world visual object recognition hard?. *PLoS computational biology*, *4*(1), e27.

Alpaydin, E. (2009). *Introduction to machine learning*. MIT press

Kriesel, D. (2018). *A Brief Introduction to Neural Networks*, available at: http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf

Mathur, A., & Foody, G. M. (2008). Multiclass and binary SVM classification: Implications for training and classification users. *IEEE Geoscience and remote sensing letters*, *5*(2), 241-245.

Support Vector Machines (2019). Retrieved from https://scikit-learn.org/stable/modules/svm.html

Bonnin, R. (2017). *Machine Learning for Developers: Uplift your regular applications with the power of statistics, analytics, and machine learning*. Packt Publishing Ltd.