CS-E4890 - Deep Learning: Project Report

Jaakko Mattila, Ilkka Saarnilehto

April 2019

1 Problem and Data Description

As our project, we chose the *iWildCam 2019 - FGVC6 Kaggle* competition. The problem focuses on identifying and categorising animals from Camera Traps (or Wild Cams). These cameras enable the collecting of large quantities of image data. This data has several possible uses, such as tracking the biodiversity and population density of the animals. Since the amount of data can be massive and the majority of images may not have animals in them, an automated recognition systems is highly useful to determine when an animal is present and to classify it. Our initial goal was beating the benchmark set in the competition and utilize generative adversarial networks (GAN) that could create more samples out of the less presented animal species and hence improve the performance of our classifier.

The amount of data provided for the problem was massive. The training set contains 196,157 images from 138 locations in South Carolina, and the test set contains 153,730 images from 100 locations in Idaho. In training data, the majority of the images has the label "empty", meaning that there is no actual animal present in the image. Additionally, out of the 23 possible labels, only 14 were actually present in the training data. The exact distribution of the labels can be seen in figure 1.

The additional information for each picture was provided in a separate CSV file, containing details such as the location and the time in the picture. The exact set of attributes provided for the images of the train dataset can be seen in 2. Test dataset had identical values without the category.

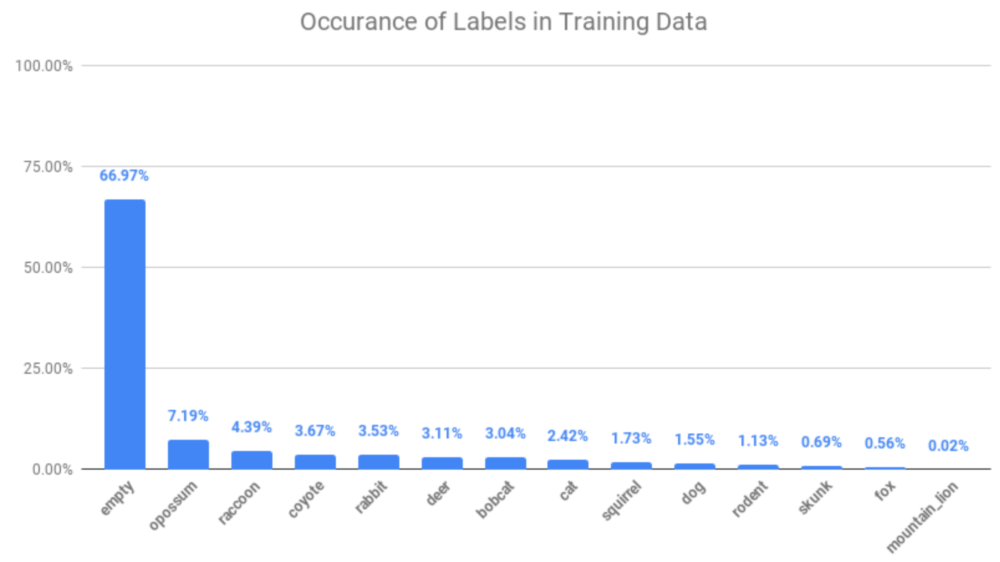
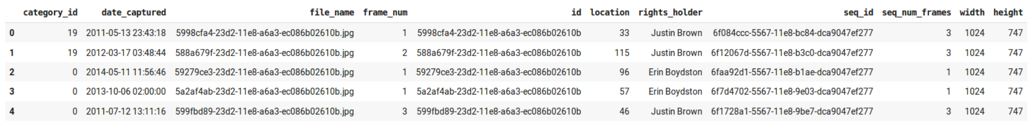


Figure 1: The percentages of different labels in the training data



	category_id	date_captured	file_name	frame_num	id	location	rights_holder	seq_id	seq_num_frames	width	height
0	19	2011-05-13 23:43:18	5998cfa4-23d2-11e8-a6a3-ec086b02610b.jpg	1	5998cfa4-23d2-11e8-a6a3-ec086b02610b	33	Justin Brown	6f084ccc-5567-11e8-bc84-dca9047ef277	3	1024	747
1	19	2012-03-17 03:48:44	588a679f-23d2-11e8-a6a3-ec086b02610b.jpg	2	588a679f-23d2-11e8-a6a3-ec086b02610b	115	Justin Brown	6f12067d-5567-11e8-b3c0-dca9047ef277	3	1024	747
2	0	2014-05-11 11:56:46	59279ce3-23d2-11e8-a6a3-ec086b02610b.jpg	1	59279ce3-23d2-11e8-a6a3-ec086b02610b	96	Erin Boydston	6faa92d1-5567-11e8-b1ae-dca9047ef277	1	1024	747
3	0	2013-10-06 02:00:00	5a2af4ab-23d2-11e8-a6a3-ec086b02610b.jpg	1	5a2af4ab-23d2-11e8-a6a3-ec086b02610b	57	Erin Boydston	6f7d4702-5567-11e8-9e03-dca9047ef277	1	1024	747
4	0	2011-07-12 13:11:16	599fbd89-23d2-11e8-a6a3-ec086b02610b.jpg	3	599fbd89-23d2-11e8-a6a3-ec086b02610b	46	Justin Brown	6f1728a1-5567-11e8-9be7-dca9047ef277	3	1024	747

Figure 2: Attributes of the images

2 Our approach

In its core the project is a simple classification problem. However the challenge is increased by several factors, such as the uneven distribution of the data, the unclear night time images and the simply massive amount of image data.

2.1 Environment

The initial decision we had to make for the project was choosing the environment setup we wanted to use for the project. We ended up with Google Colab due to having previous positive experiences with it. This is where the sheer amount of data created a lot of issues. Google Drive does not effectively handle folders that contain several thousand of files and size several Gigabytes. We however wanted our entire solution to be based on cloud and not to handle any of the data locally. We ended up countering the problem by splitting the data to sub-folders.

2.2 Preprocessing

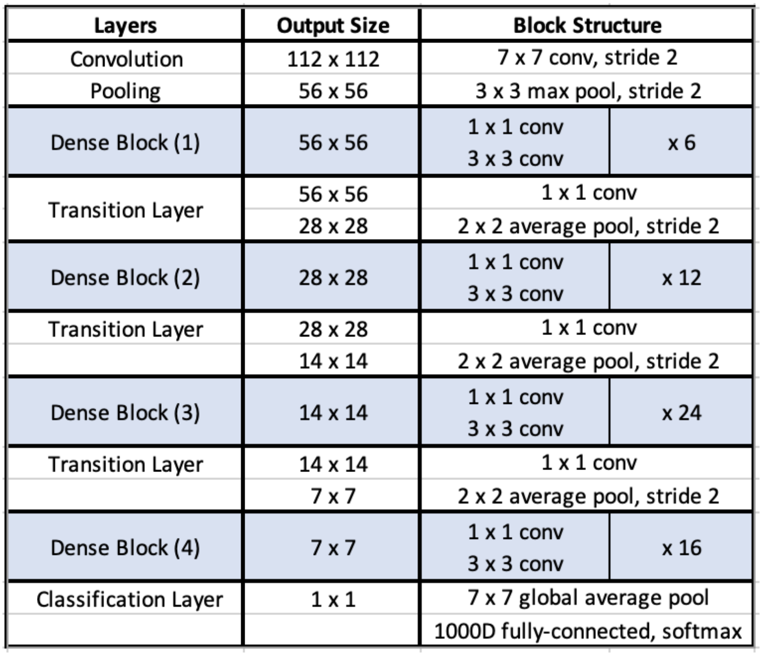
The first step was processing the data in a way that operating with it would be easier and yield more trustworthy results. First step was re-sizing. All the images were transformed to 128x128 from the original 1024 x 747. We tried even smaller resolutions, but that made identifying the smaller and unclearer animals in the images virtually impossible.

To counter the problems of the unclear night images, we used a method called CLAHE(Contrast Limited Adaptive Histogram Equalization). The method was introduced in an open kernel in Kaggle. The method aims to get more out of the dark photos including white pixels via localized contrast adjustments. It is based on computing several histograms, each of witch corresponds to a distinct section of the image . These histograms are then used to redistribute the lightness values of the image. The method was used in combination with white balancing, that attempts to retain color information as the illumination colors change. We implemented all the methods and processes for applying these modifications. However, given the time it took to just re-size the images to 128 x 128 resolution for training, we ran out of time and these processing tools were not applied for the images used in the final network.

2.3 The Model for the Classification Task

For the classification itself we used a convolutional neural network (CNN), and more precisely a deep convolutional neural network called DenseNet. It has several advantages: it alleviates the vanishing-gradient problem, strengthens feature propagation, encourages feature reuse, and reduces the number of parameters [1]. The main idea of a DenseNet is that any layer has direct connections to all subsequent layer. Traditional convolutional networks with L layers have L connections - one between each layer and its subsequent layer. The DenseNet network has $L(L+1)/2$ direct connections [1].

We also decided to use transform learning in the model due to the resource consuming nature of using massive amount of image data for the training. Transfer learning means using an existing model trained on one task is re-purposed on a second related task. In this case we used pre-weighted DenseNet-121 model that is based on ImageNet. The numeral 121 refers to the depth of the network. The exact structure of the network is described in figure 3. Each of the *conv* layers shown in the table correspond to sequence "Batch Normalization - Rectified Linear Unit - Convolution".



Layers	Output Size	Block Structure	
Convolution	112 x 112	7 x 7 conv, stride 2	
Pooling	56 x 56	3 x 3 max pool, stride 2	
Dense Block (1)	56 x 56	1 x 1 conv 3 x 3 conv	x 6
Transition Layer	56 x 56	1 x 1 conv	
	28 x 28	2 x 2 average pool, stride 2	
Dense Block (2)	28 x 28	1 x 1 conv 3 x 3 conv	x 12
Transition Layer	28 x 28	1 x 1 conv	
	14 x 14	2 x 2 average pool, stride 2	
Dense Block (3)	14 x 14	1 x 1 conv 3 x 3 conv	x 24
Transition Layer	14 x 14	1 x 1 conv	
	7 x 7	2 x 2 average pool, stride 2	
Dense Block (4)	7 x 7	1 x 1 conv 3 x 3 conv	x 16
Classification Layer	1 x 1	7 x 7 global average pool	
		1000D fully-connected, softmax	

Figure 3: DenseNet-121 architectures for ImageNet

The loss function used was binary cross entropy with sigmoid activation function.

3 Experiments and results

The main tool we used to track our result was a macro F1 score which was also used to evaluate the results in Kaggle. F1 is calculated for each label, and the Kaggle's final score is the unweighted mean of all labels F1 scores.

Due to the difficulties of handling the massive original dataset, we started the training with significantly smaller portions of data. As a starting point we used only 4000 images, from 4 subfolders of 1000 images in each, for the training. For this training attempt, we used 6 epochs and batch size of 100. After that we incrementally increased the amount of training data.

Similarly the test data faced challenges in processing so the first submission to Kaggle could only evaluate 22000 images. With this submission, we received score of 0.090 which is still far below baseline of 0.125. As we had only a little time after this submission, our effort went to increase the training data to 79000 images and the test sample for evaluation could be increased to 56103, around 1/3 of the whole test dataset.

However, even with limited amounts of data the model proved to be efficient. This is illustrated in figure 4. Each epoch improves the results.

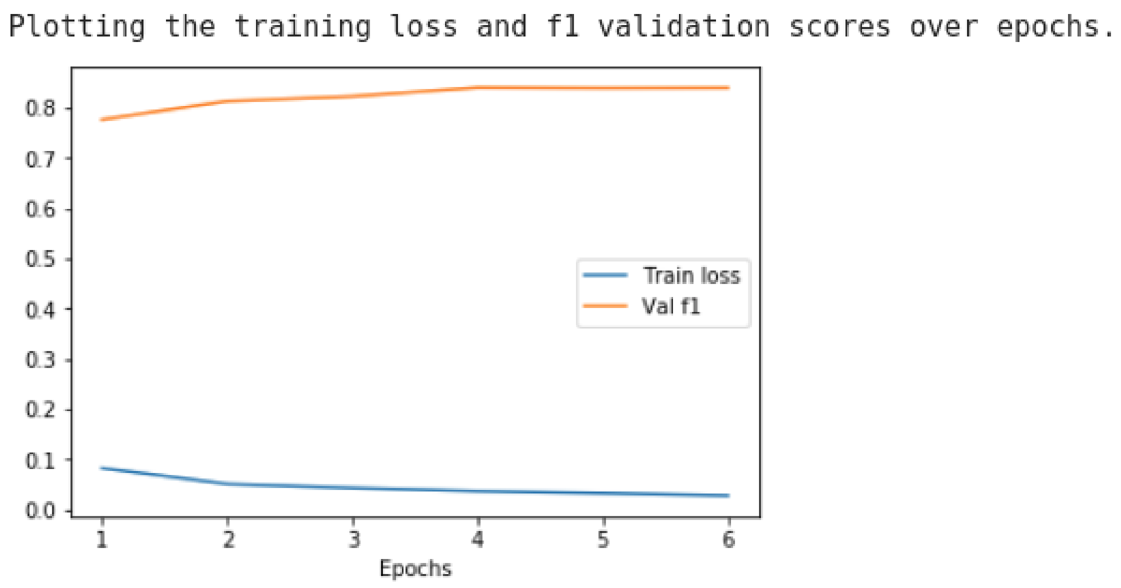

Plotting the training loss and f1 validation scores over epochs.

Figure 4: The model in action

Figure 5 shows the very first run we did with the model. It illustrates the upside of transfer learning. At the very beginning the model already does well.

3.1 Software and computation tools

Our GitHub repository includes the source code as well as the description of our technologies and open kernels used in the `README`.

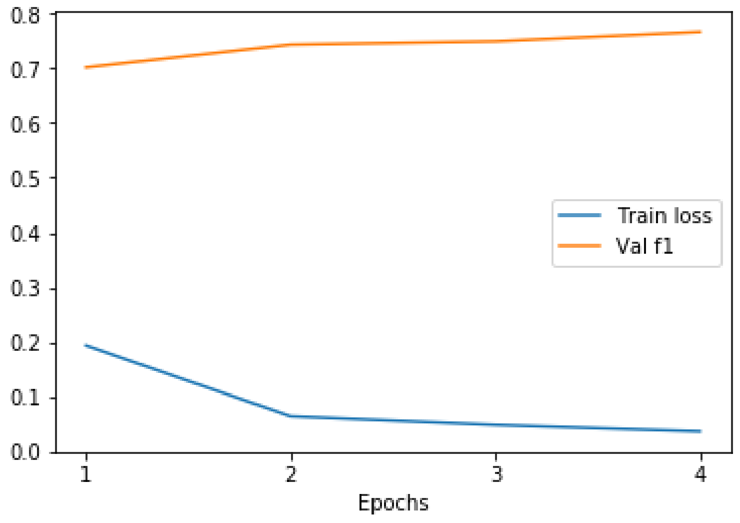


Figure 5: The first run

3.2 Data

The data of around 43 GB and over 150 thousand images per folder turned out to prove very challenging to manage with Google Drive. Especially, as any time we tried to access an image to read it, the request could easily timeout if the folder it was in had too many files to browse. Sometimes this could be overcome be just repeating the request and ignoring the timeout error. However, we decided to follow advice in the Internet to this and divided our images into subfolders of 1000 images each. This turned out to be also very time consuming process and required also supporting code to find the images that were scattered across these folders. However, it enabled us to work with the data and get some results.

4 Conclusion

While the original goal of the project was scoring as overcoming the Kaggle benchmark using GAN based net, during the course of the project the focus shifted just being able to handle the large dataset in a cloud environment and getting something functional out of the transfer learning net. Overall, we managed to get the whole pipeline working despite great challenges in the data storage and accessing. The model peformed extremely well even with the limited data we had available for it.

The solution could easily be improved, but we simply lacked the time to do it. One such idea would have been to categorical cross entropy instead of binary one as there were more class outputs than two and it would have likely been more robust to imbalanced dataset.

References

[1] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. 2016.