Machine Learning in Practice

Team Report - Classifying Nemo
The Nature Conservancy Fisheries Monitoring Competition

Christoph Schmidl
s4226887
c.schmidl@student.ru.nl

Denis Pogosov
s4750276
denis.b.pogosov@gmail.com

Emma Valtersson
E711929
emma.valtersson@mpi.nl

Lars Kuijpers
s4356314
ljt.kuijpers@student.ru.nl

Lisa Boonstra
s3018547
l.boonstra@student.ru.nl

April 19, 2017

1 Problem description

In the Kaggle competition "The Nature Conservancy Fisheries Monitoring", we faced a problem of classifying a data set with photos from cameras mounted on different boats into eight different classes (6 fish species of interest, 1 with other fish species, 1 with no fish). The data set posed several problems for our machine learning task. The classification of the fish species was made more difficult by the fact that the image quality differed; some photos seemed to be taken with a small time interval, which resulted in correlations in the training data; the photos were taken at different times of the day (leading to colour distortion); the fish were sometimes difficult to localize; two classes were very similar (ALB and BET) and the amount of images differed greatly across classes (ranging from 67 to 1719). In addition, the two separate stages of the competition differed in that the second test set had more images per class, and the similarity across images was greater (e.g., different boats).

2 Approach

In order to overcome difficulties with the training set, we began by collecting additional images from the Kaggle discussion thread about extra data [1], mostly from ImageNet [3]. From these images, we attempted to exclude the duplicates and images with a different fish species. To create the final training set, we combined the external internet and Kaggle images for which we augmented the number of images per class (see Table 1). The augmentation for class balance was done using horizontal flipping, 90 degree rotation counterclockwise, cropping out corners, shifting, white noise addition, and scaling. As a second step of the data preparation, in order to overcome difficulties to localize the fishes, we annotated bounding boxes around the region of interest (the fish). For the Kaggle training images, we found the annotations posted on the discussion forum by another user.

Since it was sometimes difficult to localise the fish in the images, we chose to work with the YOLO2 framework that has been documented with fast performance on object detection, with reported high accuracy [2]. It trains a fully-convolutional neural network on the entire labelled images (with bounding boxes). Before feeding the images to the network we resized the images such that they were all 320 by 320, the size we used in YOLO2. For our full implementation of this framework, please consult the following github repository [4].

	ALB	BET	DOL	LAG	SHARK	YFT	OTHER	TOTAL
Original	1719	200	117	67	176	734	299	3312
Original augmented	1719	1600	936	1072	1408	1468	1196	9399
External data	87	251	457	84	154	436	0	1469
External augmented	1408	2008	1828	1072	1232	1744	0	9292
Final data	3525	3608	1510	2144	2640	3212	1196	**17835**

Table 1: Amount of images after adding extra data.

3 Results

We attempted several early provisional submissions, starting by using a Kernel submission from the Kaggle forum that ended up below the sample submission. After this we improved our ranking with a model using transfer learning from the VGG-16 network with batch normalisation, resulting in 280th place (score of 0.99059). After this, we improved even further using the YOLO2 approach, ultimately ending up in the 10th place, with a score of 0.51378.
For the second stage of the competition, we submitted the two most well-performing models that we could use for our final submission. When the private scores were revealed, we dropped down to the 256th position with a score of 2.06869.

4 Discussion

In general, our submissions in the first stage of the competition yielded very good results, with a high ranking on the leaderboard. However, the two models that we submitted for the second stage of the competition had the best public score, but were probably underfitted on the training data. This is a consequence of too few training images.The underfitting would be a valid explanation for the drop on the leaderboard, and the much lower private score. The reason being that our networks were not able to identify distinctive features for the different classes. Since the test images in the second stage were different from the ones in the first stage, the training data could not account for these differences.

Another issue was caused by the external data set that we extracted from online sources. Those images were quite different from the data that Kaggle provided in their training data set with fishes held up and shown off by the fisherman, or swimming in the sea. However, we did see an improvement in our score before and after using the external data. It is nevertheless important to note that we did notice erroneous labels in this data set, which suggests that external data needs to be thoroughly examined before involving it in the training phase. This leads to an additional problem, because (for this particular competition) it was very hard to distinguish certain classes (i.e., ALB and BET).

For future image classifications, we envision that more data augmentation techniques (such as small angle rotation, colour/contrast/saturation jittering, more crops, etc) help to overcome the problem of a small data set without enough image variation for a good neural network. This would potentially also counteract the problem of model overfitting, since a more varied input creates a more diverse neural network. Moreover synthetic data can be helpful in the case of correlated input data and if training set does not represent objects properly (e.g. photos taken only from particular angles). Furthermore, an ensemble model that incorporates a few different models may perform better than a single model, since this combined model would better predict a more diverse test set. This is due to the different specializations from the individual models. Alternative to ensemble model is detection on augmented test images, that we noticed improved overall classification.

5 References

[1] `http://www.kaggle.com/c/the-nature-conservancy-fisheries-monitoring/discussion/25428`
[2] Joseph Redmon et al. (2016). YOLO9000: Better, Faster, Stronger. CoRR, abs/1612.08242
[3] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). IEEE.
[4] `https://github.com/dpogosov/yolo_kfm`

Appendix

Individual Contributions

Christoph

- Worked with the SurfSara cluster and tried different, small models to get acquainted with the whole SurfSara ecosystem. The SurfSara cluster helped a lot as soon as you were sure that the whole model pipeline worked. It was rather difficult to use it for mere prototyping use cases. For the purpose of prototyping I had to configure my desktop pc at home.
- Configured Desktop PC at home for dual booting Windows 10 with Ubuntu 16.04 because a virtual environment was not suited for GPU operations. The whole configuration process for Ubuntu 16.04 with Keras and Thenao as its backend has been written down in another document. Different problems and their fixes regarding memory issues on a linux environment has been written down as well.
- Worked through tutorials for Keras and incorporated different models found on the Kaggle forum. Keras seemed a good choice to generate the first submissions with ease. Keras was also used by most of the users in this competition, therefore it was possible to find many ideas on the forum and later on tweak them to ones own need.
- One specific tutorial for Keras was tailored towards image classification and was used for later optimization (`https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html`)
- Worked on a neural network, making use of transfer learning, using a pretrained VGG-16 network with batch normalization using Keras and Theano. Gained good results (place 280 on the public leaderboard).
- Worked through the whole Fast.Ai course regarding CNNs (`http://course.fast.ai/index.html`) and incorporated ideas like pseudo labeling which can be used for the next competition.
- Did the 5-minute presentation
- Github repository: `https://github.com/ChristophSchmidl/machine-learning-in-practice`

Denis (total about 200 hours of work)

- Looked for an approach to solve the Fisheries monitoring competition. I found and learned the approaches of region proposal convolutional network, such as variations of R-CNN (generic, Fast and Faster R-CNN), as well as Yolo v2. Moreover, I looked for approaches, which were not demanding to computational resources, such as Enet. I decided to work with Yolo v2, because of reported (by authors) high accuracy and speed. Moreover, it does two stage detection: finds an object on image and classify it.
- Prepared my desktop PC for Yolo installation: installed MS Visual Studio 2015, CUDA, OpenCV, Tensorflow and Darknet frameworks. Unfortunately CNNs need a good GPU, which I do not have. CPU detection was very slow, and estimated training time of Yolo, based on non-augmented dataset was 2+ months.
- Worked with SURFsara cluster to train Yolo. Arranged installation of Darknet framework on the cluster and run training pipeline. After two epochs I noticed very good improvement of classification from less than 5% to approx. 40% accuracy. Using SURFsara for a lot of experiments with deep neural networks is expensive and exceeds the given budget of 2000 sbus.
- To arrange confunction with Kaggle, made two scripts in matlab: first converts json files (annotations) to Yolo format; second converts Darkflow output json files to csv for Kaggle and incorporates fine tuning of likelihoods.
- Asked our coach (Twan van Laarhoven) to give us access to a university GPU server "Speedy" (not limited by any budget). Worked with Speedy server to train Yolo. There was no direct access, so I made several scripts to transfer command and files via intermediary server lilo. Arranged installation of Darknet framework on the server and run training pipeline.

- Tried different ways to train Yolo and prevent overfitting. Arranged about 50 hours and at least 10 pipelines of training with different parameters. Came to conclusion that the given training set was essentially insufficient.
- Tried and checked efficiency of different ways to fine tune likelihoods:
  - Clipping (limit likelihoods by a number, e.g. 0.95)
  - Gain (multiply all the likelihoods by a number, e.g 1.2)
  - Use the rule for non-mutually exclusive events (to improve likelihoods for several objects)
- There was a class where no fish detected (NoF) that is managed by likelihood threshold. However, some images were hard to be classified even by a human expert, so I had to use some approaches for that class:
  - give a number for NoF class (e.g 0.5) and use uniform distribution for the rest
  - use real distribution of classes among a test set. This minimize entropy and should give lower error.
- Created a script in Matlab that estimated the accuracy from the error given by Kaggle. The best accuracy was 70%. Hence, asked team members to find and annotate extra images.
- Prepared extra images for testing. Created scripts in Matlab that augmented images (given and extra) by horizontal flipping, 90 degree rotation counterclockwise, cropping out corners, shifting, white noise addition. Scaling was incorporated in Darknet framework.
- Arranged about 40 hours and about 5 pipelines of training with different parameters on augmented data.
- Found a very good early stopping point and set of parameters for fine likelihoods tuning. The error was 0.51 and it gave us 10th place on the public leaderboard. The Yolo was checked for overfitting with a validation set.
- According to Kaggle rules, before we could see the second stage test set, I prepared and uploaded the two models (only two were allowed) and the whole dataset for submission to Kaggle.
- Arranged testing the second stage dataset and submitted predictions. According to the rules we were not allowed to change anything from the previously submitted models.
- The second stage error was 2.06 that was higher than the error at the first stage. I see two major reasons: 1) insufficient amount of (augmented) training images, 2) tightly tuned likelihoods (e.g. change gain from 1.2 to 0.75 gives the error amount of 1.68).
- Due to the fact that Yolo worked well on the first stage test set and it worked not so efficient on the second stage test set I decided to check 3 following ideas.
  1. Continue training of Yolo and check the accuracy of the selected early stopping point. I arranged also 20 hours of training and noticed that the error is slightly lower after all. Hence, the early stopping point was not perfect. However, it was not possible to see it on the validation set I had had and another performance parameters I had checked.
  2. Pseudo-labelling of the second stage dataset. I created a matlab script for that and arranged training. This technique is very sensitive to parameters. I did not program it properly (needed extra 40 hours), as a consequence it did not bring much benefits. Note that it was not allowed to use the second stage set for training.
  3. Augmented the test set and picked up the most likely prediction among the augmented copies of an image (greedy algorithm). I created two matlab scripts for that and arranged testing on 3 models. Surprisingly, it improved error much and error was 1.59 that could be the best result (among the course peers). Note that the second stage images were not used for training (e.g. by pseudo-labelling).
- Made the presentation (would be presented on 03th May of 2017).
- Made a Github repository with guidances and all the relevant files to training, testing and supplementary operations (`https://github.com/dpogosov/yolo_kfm`)

Emma

- Tried implementing different neural networks using different tutorials
  - `https://chsasank.github.io/keras-tutorial.html`
  - `https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html`
- Collected images from external data sources.
- Annotated external images with bounding boxes.
- Helped with the preparation of the first 5-minute presentation.
- Helped writing the final report.

Lars

- Did exploratory work on data and approaches from other people.
- Created sample submission using model from the discussions.
- Tried implementing neural networks using Tensorflow. Struggled with running it because of many issues with required libraries.
- Helped writing the final report.

Lisa

- Did tutorials regarding Tensorflow
- Tried to get Emmas network to work. Ultimately couldnt because of unovercomable incompatibilities with the python libraries.
- Tried to work on the SurfSara cluster.
- Annotated external images with bounding boxes.
- Helped with the preparation of the first 5-minute presentation.
- Helped writing the final report.