Parallel Implementation of Data Augmentation for Alignment-Free Facial Attribute Classification in PyTorch

Master Project

Team: Noah Chavannes and Yves Rutishauser

Supervisor: Prof. Dr. Manuel Günther

1 Introduction

Typically, deep neural networks require lots of training data in order to learn their tasks. For smaller-size training sets, the amount of training data is often artificially increased by some form of data augmentation, i.e., input images are scaled, mirrored and cropped differently. However, data augmentation can also be used to make the networks robust to some input variations that appear in natural data. In an approach to make, for example, facial attribute classification more robust to facial misalignment, we have shown [Günther et al., 2017] that extended data augmentation including rotation, scaling and blurring of training faces enabled the network to waive the otherwise required face alignment step. Additionally, it was shown that data augmentation during testing can improve recognition accuracies considerably.

To enable facial attributes to be classified without face alignment, a two-tier training strategy was implemented. While the network training used the GPU to accelerate, the data augmentation was performed massively parallel on the CPU. For this, we had implemented a special input layer in the Caffe framework [Jia et al., 2014] where we achieved the massive parallel image processing using a C++-implementation with OpenMP. After training a ResNet-50 network topology on the training set of the CelebA dataset [Jia et al., 2014] and evaluating it on the test set, we could show that we did not need to rely on the alignment of faces, but we could use the bounding box of the face detector directly to crop faces from the image without loss in classification accuracy. Unfortunately, Caffe is outdated by now and has been superseded by other deep learning frameworks such as PyTorch [Paszke et al., 2019], so the implementation is no longer working.

2 Assignment

The assignment in this master project is the implementation of the image preprocessing / data augmentation technique in PyTorch. For an incoming face images including its facial landmarks, an image transformation has to be implemented that computes a bounding box from the landmarks. Additionally, random factors such as a random scaling, rotation and shifting of the bounding box needs to be applied. With this perturbed bounding box, the face region must be cropped from the original image, and additional manipulations such as blurring, noising, color adaptations or alike, need to be applied. Finally, the image needs to be transformed to match the requirements of the target network (input resolution, pixel scale, ...). More details can be found in [Günther et al., 2017]. For the implementation of the image processing, the image processing library Bob [Anjos et al., 2012] and, particularly, its image processing libraries[1] should be used. As this library is only available under Linux, and as our deep learning servers use the Linux operating systems, the students need to make themselves familiar on how to set up virtual environments using Conda and run experiments under Linux.

Deep neural networks use batches of inputs to train a step, so a batch of inputs must be available to continue the training process. Since the image processing might be too slow when run in sequential mode, the data manipulation technique must be applied in parallel on many different facial images and their respective facial landmarks. As python includes a Global Interpreter Lock (GIL) that disallows running several threads in parallel,[2] an implementation with multiple processes is required, where inter-process communication needs to be designed carefully. Additionally, requirements such as dataset shuffling after each training epoch needs to be implemented, where special attention is required since images, facial landmarks and the target attributes

[1] `http://www.idiap.ch/software/bob/docs/bob/docs/stable/bob/bob.ip.base/doc/guide.html`

[2] Python has an implementation for multithreading, but it does not allow to run multiple of these threads at the same time in parallel.

need to be shuffled in the same way. Finally, the parallel data augmentation needs to be incorporated into the training procedure, which needs to be developed and implemented, too.

After all parts of the processing chain have been implemented, the AFFACT network [Günther et al., 2017] should be re-trained in PyTorch. Experiments of the paper should be reproduced, especially the classification of facial attributes should arrive at about the same level as reported in the paper. For this purpose, the evaluation strategy needs to be implemented.

If time allows, the very same image preprocessing technique should be used to train a face recognition network, for example, by modifying an existing face recognition training framework.[3] Face recognition capabilities should be evaluated with aligned and unaligned images, and compared with a network trained without data augmentation.

3 Schedule

Assuming 30 hours of work per week and a total of 18 ECTS with an average of 30 hours per ETCS, we arrive at a total workload of 18 weeks. These should be distributed as follows. Since the project is handled by two students, it should be possible to work on Milestones 2 and 3 in parallel. Also, Milestone 4 can be split between the two students, i.e., one is training the deep network while the other is running the face detector on the test set.

Week 1-2 Setting up the work environment, installing all required tools, implementing the core training procedure in PyTorch and getting familiar with the image processing tools.

Week 3-4 Training a deep network with the desired topology on the pre-aligned images provided by the CelabA dataset and implementing an evaluation method.

⇒ Milestone 1: A deep facial attribute classification network is trained and evaluated on the pre-aligned images.

Week 5-8 Implementation, testing and documentation of the image processing / data augmentation techniques.

⇒ Milestone 2: A randomly perturbed face image in the desired image resolution is created based on an input image and its facial landmarks.

Week 9-12 Implementation, testing and documentation of the parallel processing scheme including data shuffling.

⇒ Milestone 3: Batches of randomly perturbed images are generated in parallel for several training epochs.

Week 13-16 Training of the network with the data augmentation technique, running a face detector on the test set images and crop the faces, and evaluate of the performance of this network on the cropped faces.

⇒ Milestone 4: The network is trained and is more stable to misalignment than the network from Milestone 1.

If time allows Training of a face recognition network on the VGGFace2 dataset [Cao et al., 2018] using the data augmentation strategy and evaluate the face recognition capability on the Labeled Faces in the Wild dataset [Huang et al., 2007].

⇒ Milestone 5: Face recognition works comparably well or better for our data augmentation technique as compared to regular training.

Week 17-18 Preparation of the Presentation and writing of the final report.

Milestones 2 and 3 need to be delivered by the students at the end of the project. Milestones 1 and 4 require the hardware for deep learning to be available and might be skipped if the resources are not available. Milestone 5 is optional.

[3]`http://github.com/grib0ed0v/face_recognition.pytorch`

4 References

[Anjos et al., 2012] Anjos, A., El-Shafey, L., Wallace, R., Günther, M., McCool, C., and Marcel, S. (2012). Bob: a free signal processing and machine learning toolbox for researchers. In *ACM international conference on Multimedia*, pages 1449–1452.

[Cao et al., 2018] Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. (2018). VGGFace2: A dataset for recognising faces across pose and age. In *Automatic Face & Gesture Recognition (FG)*. IEEE.

[Günther et al., 2017] Günther, M., Rozsa, A., and Boult, T. E. (2017). AFFACT: Alignment-free facial attribute classification technique. In *International Joint Conference on Biometrics (IJCB)*, pages 90–99. IEEE.

[Huang et al., 2007] Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, Univ. of Massachusetts, Amherst.

[Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM International Conference on Multimedia*, pages 675–678. ACM.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeuRIPS)*.