Project B
ECE1512 Digital Image Processing and Applications
Department of Electrical and Computer Engineering
University of Toronto
Winter 2021

Deadline: Monday, April 5, 5:00 PM EST.
Teaching Assistant: Karush Suri (karush.suri@mail.utoronto.ca: Please use **"Project-B"** as the subject line.)
This project provides a FAQs page on Quercus. Please monitor it regularly for updates.

Introduction

Modern image processing techniques in the era of machine learning have undergone radical shifts. The field of Explainable Artificial Intelligence (XAI) [1] has allowed vision detection pipelines to incorporate complex architectures capable of real world applications [2]. The project will center its discussion towards these applications and utilize Convolutional Neural Networks (CNNs) for implementation.
The goal of this project is to equip you with the tools and technologies required for developing an end-to-end image recognition pipeline. Specifically, the project will focus on the setting of Facial Recognition and extend it to retrieve meaningful contexts such as emotions and Facial Action Units (FAUs) [3, 4, 5]. The project is divided into 2 tasks, (1) detection of emotions from images consiting of facial expressions and (2) detection of FAUs from images consisting of varying degree of emotions.

Submission Guidlines

Similar to Project A, a project report consisting of a maximum of 20 pages in IEEE style must be submitted on Quercus before the project deadline. Only one report per group needs to be submitted. **Late submissions will not be accepted.** Submission of supplementary material (code base, presentation slides, poster, etc.) is not mandatory but **highly encouraged** as a single `supplementary.zip` file initialized with a `README` document.

Preliminaries

Programming Requirements

This project relies on Python. You will program detection pipelines using the PyTorch package. If you are not familiar with PyTorch then here is a list of resources which will get you started on the assignment. The content provided here is sufficient for you to complete the project. However, if you wish to learn PyTorch in detail then please refer to other extensive sources [6].

- ECE1512 PyTorch Tutorial
- PyTorch documentation

GPU Requirements

Note that **none of the tasks require the use of a GPU** and you should be able to train your models on local machines. However, you can use a GPU by accessing Google Colab. Following are the steps to enable a GPU using Colab-
1. Upload the code base to Colab using your Google Drive.
2. Navigate to `Runtime->Change runtime type` in the top bar.
3. Change runtime accelerator to `GPU` and click `Save`.
4. Use `device = torch.device("cuda:0")` after importing libraries and call `.to(device)` function to transfer your model and tensors to GPU. Make sure that model and tensor both are placed on GPU.

Task 1: Facial Emotion Detection (15 pts)

In this task you will detect emotions from static facial images. More specifically, you will implement the Xception [7] architecture for detecting emotions and understand its efficacy as an image recognition pipeline. You will make use of the Facial Expression Recognition (FER2013) [8] dataset. The dataset consists of 35,685 examples of 48x48 pixel grayscale images of faces. Out of these, 28,710 comprise of the training set with the remaining 6,975 forming the test set. You can read more about the dataset here. Images are categorized based on the emotion shown in the facial expressions (happiness, neutral, sadness, anger, surprise, disgust, fear).

Download the code for this task from the Quercus webpage. Package will be downloaded as `Task-1.zip`. Extract the folder to obtain the directory structure consisting of the following folders-
`dataset`- The folder contains FER2013 dataset as a single `.csv` file.
`src`- The folder contains `train.py`, `model.py` and `check.py`. You will use these files to implement and improve the Xception model.
`test`- The folder contains a set of 4 test cases.
`test2`- The folder contains a different set of 4 test cases.
Install the required libraries listed in `requirements.txt` using `pip install -r requirements.txt`.

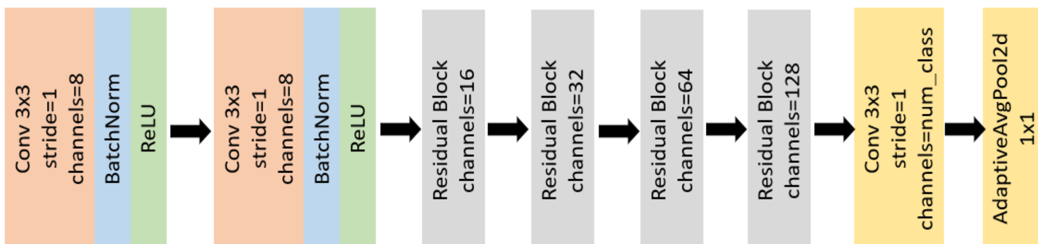

Figure 1: The Xception model architecture

1. Figure 1 presents the Xception model for emotion recognition on FER2013 dataset. Implement Figure 1 in `model.py` using PyTorch. The model class along with the required blocks has already been implemented for you. Write your program in the `init` and `forward` functions of the class. (1 pt)
2. Complete functions `plot_loss()` and `plot_acc()` in `train.py` for plotting loss and accuracy metrics. (1 pt)
3. Train the model for 20 epochs by running `train.py` file (`python train.py`). This should take approximately 16 minutes of wall-clock time on your local CPU machine. You will observe two kinds of metrics while training - public and private metrics. These are based on data samples which are made available to the model. Upon completion of training, report training accuracy, training loss, private test accuracy, private test loss, plot for loss curve and plot for accuracy curve. (2 pts)
4. You will now evaluate the model and gain intuition towards the representations learned by the separable convolutional layers of Xception architecture. The `test` folder contains 4 test images named after their true labels. Run `check.py` file (`python check.py`). Open and report the file `guided_gradcam.jpg`. Observe the model predictions and activations corresponding to test images. Can you comment on the performance of the model? (2 pts)
5. Replace the contents of `test` folder with that of `test2` folder. Now run `check.py`. Open and report the file `guided_gradcam.jpg`. What happened to the model predictions and activations? Why? (3 pts)
6. Using **only one** of (a) *finetuning*, (b) *transfer-learning* or (c) *class-reweighting*, conduct a small experiment to address the limitations of the model observed above. Which method did you select? Why? (2 pts)
7. How does the selected method improve the model? Explain your approach and its limitations in detail. Note that the emphasis is not on the improvement of metrics but on implementing your ideas in order to address the limitations of the model. (4 pts)

Task 2: Facial Action Unit Detection (25 pts)

In this task you will detect Facial Action Units (FAUs) [9] from static facial images consisting of varying level of emotions. FAUs are specific regions of the face which allow one to estimate the intensity [10] and type of emotion one is experiencing in a given image. Utility of FAUs has seen a tremendous adoption in research [11, 12, 10] and animation [13] as a result of their unique action coding system [9]. Each facial movement in the FAU coding system is encoded by its appearance on the face. For instance, consider Figure 2.

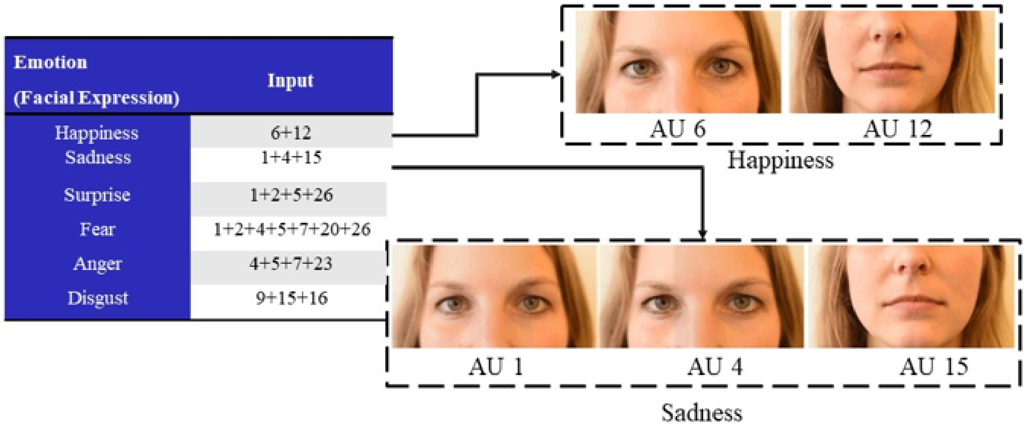

Figure 2: Combination of FAUs yields facial emotions

The AU code for *cheek raise* (AU6) and *lip corner pull* (AU12) when combined together give the *happiness* (AU6+12) emotion. As another example, the AU codes *inner brow raise* (AU1), *brow low* (AU4) and *lip corner depress* (AU15) together give the *sadness* (AU1+4+15) emotion. AU codes start from 0 and end at 28 with the 0^{th} code corresponding to *neutral face*.

While there are various FAUs present in the Facial Action Coding (FAC) system, most applications make use of only a few of them to characterize emotions. These FAUs are commonly referred to as Main FAUs and are presented in Figure 3. A complete list of AU codes and their corresponding facial expressions can be found on Wikipedia.

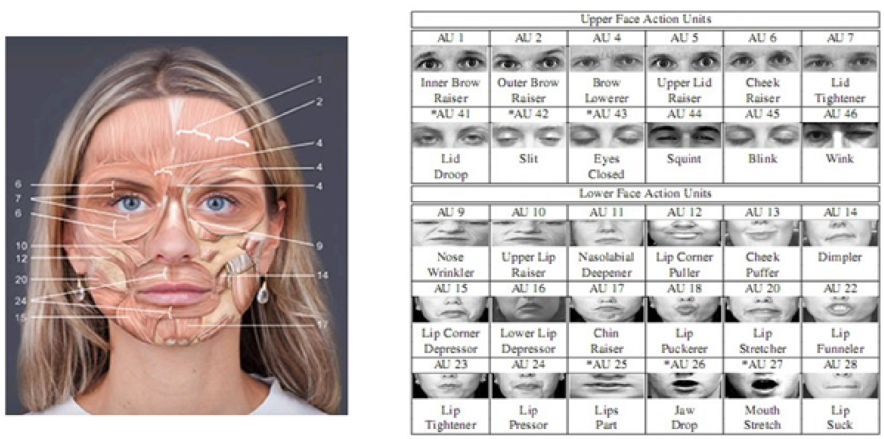

Figure 3: Commonly used FAUs from the FAC system.

You will utilize FAUs in this task to estimate the intensity of facial expressions and try to link them with their corresponding emotions. However, you will do so without making use of a dataset and only use a handful of images from the FFHQ dataset [14] (often referred to as few-shot tuning). For this purpose, you will adopt a slightly modified version of the deep ResNet [15] architecture which is pretrained on the BP4D FAUs dataset [16].

Download the code for this task from the Quercus webpage. Package will be downloaded as `Task-2.zip`. Extract the folder to obtain the directory structure consisting of the following folders-
`src`- The folder contains `train.py`, `network.py` and `run_demo.py`. You will use these to write the program.
`train`- The folder contains a handful of sample images for finetuning the model.
`test`- The folder contains a set of 5 test cases.
`test2`- The folder contains a different set of 5 test cases.

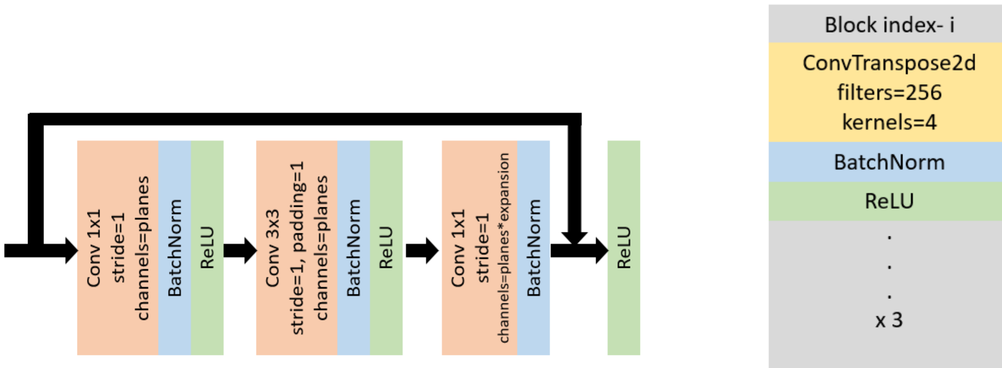

Figure 4: Left: ResNet bottleneck block, Right: Deconvolutional block

1. Figure 4 (left) presents modified the ResNet architecture consisting of bottleneck residual blocks and deconvolutional layers. Unlike Task-1, the bottleneck block for this task has not been implemented. Implement the bottleneck block in the `BottleNeck` class by completing the `init` and `forward` functions. The variables have already been created for you. Note that the BatchNorm layer will take an additional *momentum* parameter. (2 pts)
2. The ResNet module is contained in the `ResNet` class which consists of `_make_deconv_layer` function. The `_make_deconv_layer` constructs a deconvolutional block by omitting the residual connections and replacing convolutions with deconvolutions (`ConvTranspose2d`). Figure 4 (right) presents a single deconvolutional block consisting of 3 sub-blocks each having `ConvTranspose2d`, BatchNorm and ReLU layers. Implement the deconvolutional blocks in the `init` function using the `_make_deconv_layer` function. The variables have already been created for you. (1 pt)
3. Execute `run_demo.py` (`python run_demo.py`). This will create the `visualize` folder with a total of 15 images (5 FAU images for each of the 3 subjects). Report the FAU intensity values corresponding to AU17. How do intensity values vary across subjects? How do they vary across FAUs? (2 pts)
4. Replace the contents of `test` folder with that of `test2` folder. Now execute `run_demo.py`. How do intensity values vary across the new subjects? What about the variation across FAUs? (3 pts)
5. Compare emotion recognition in the presence of FAUs with the naive emotion recognition approach taken in Task-1. How and why do the FAUs play a role in better interpretation of emotions? (3 pts)
6. You will now try to improve the model by only using a handful of sample images. The program to train the ResNet architecture is provided in `train.py`. You are required to implement the main training of the loop by zeroing the gradient, differentiating the loss and taking a gradient step using the optimizer. Implement this training procedure in `train.py` and train the model for 5 epochs. This should take approximately 10 minutes of wallclock time on your local CPU machine. Report the final training loss. (2 pts)
7. Change the model path to `tuned_model.pth` and execute `run_demo.py` on the contents of `test2` folder. Does the few-shot finetuning help in estimating intensities of FAUs on the new subjects. If yes, then why? If no, then why not? (2 pts)
8. Now that you have had a chance to implement, understand and train the end-to-end ResNet framework, you will improve the performance of the model with regards to its limitations in effectively estimating FAU intensities. Read **only one** of the following four papers- [17, 18, 19, 20]. What is the main novel idea of the paper? (2 pts)
9. How does the proposed method improve FAU estimation in comparison to the above ResNet model? (2 pts)
10. What are some of the limitations of the proposed method? How can they be addressed? (2 pts)
11. Conduct a small experiment to implement the novel component of the proposed method with the ResNet architecture. Evaluate the modified model on the same test cases from `test` and `test2` folders. Explain your approach and its outcomes in detail. (4 pts)

References

[1] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[2] Sam Greydanus. Scaling down deep learning, 2020.

[3] Junya Saito and Kentaro Murase. Action units recognition with pairwise deep architecture. *arXiv preprint arXiv:2010.00288*, 2020.

[4] Didan Deng, Zhaokang Chen, and Bertram E Shi. Multitask emotion recognition with incomplete labels. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)(FG)*, pages 828–835.

[5] Felix Kuhnke, Lars Rumberg, and Jörn Ostermann. Two-stream aural-visual affect analysis in the wild. *arXiv preprint arXiv:2002.03399*, 2020.

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037. Curran Associates, Inc., 2019.

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[8] Ian J Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *International conference on neural information processing*, pages 117–124. Springer, 2013.

[9] Paul Ekman. Facial action coding system (facs). *A human face*, 2002.

[10] Zhiwen Shao, Zhilei Liu, Jianfei Cai, Yunsheng Wu, and Lizhuang Ma. Facial action unit detection using attention and relation learning. *IEEE transactions on affective computing*, 2019.

[11] Michel Valstar and Maja Pantic. Fully automatic facial action unit detection and temporal analysis. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 149–149. IEEE, 2006.

[12] Wen-Sheng Chu, Fernando De la Torre, and Jeffery F Cohn. Selective transfer machine for personalized facial action unit detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3515–3522, 2013.

[13] Prem Kalra, Angelo Mangili, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of facial muscle actions based on rational free form deformations. In *Computer Graphics Forum*, volume 11, pages 59–69. Wiley Online Library, 1992.

[14] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Xing Zhang, Lijun Yin, Jeffrey F. Cohn, Shaun Canavan, Michael Reale, Andy Horowitz, Peng Liu, and Jeffrey M. Girard. Bp4d-spontaneous: a high-resolution spontaneous 3d dynamic facial expression database. *Image and Vision Computing*, 32(10):692–706, 2014.

[17] Itir Onal Ertugrul, Le Yang, László A Jeni, and Jeffrey F Cohn. D-pattnet: Dynamic patch-attentive deep network for action unit detection. *Frontiers in computer science*, 1:11, 2019. https://www.jeffcohn.net/wp-content/uploads/2019/07/BMVC2019_PAttNet.pdf.pdf.

[18] Lezi Wang, Chongyang Bai, Maksim Bolonkin, Judee Burgoon, Norah Dunbar, VS Subrahmanian, and Dimitris N Metaxas. Attention-based facial behavior analytics in social communication. In *30th British Machine Vision Conference, BMVC 2019*, 2020. https://bmvc2019.org/wp-content/uploads/papers/0491-paper.pdf.

[19] Andrei Racoviteanu, Mihai-Sorin Badea, Corneliu Florea, Laura Florea, and Constantin Vertan. Large margin loss for learning facial movements from pseudo-emotions. https://bmvc2019.org/wp-content/uploads/papers/0498-paper.pdf.

[20] Koichiro Niinuma, László A. Jeni, Itir Önal Ertugrul, and Jeffrey F. Cohn. Unmasking the devil in the details: What works for deep facial action coding? In *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*, page 4, 2019. https://bmvc2019.org/wp-content/uploads/papers/0403-paper.pdf.

*****