Laboratory 13: TensorFlow image classification

In this laboratory exercise, you will use TensorFlow to train an efficient machine learning model to recognize the 10 native North American species of *Acer* (maples)[1] from herbarium specimen images. The images been processed to redact the specimen labels with opaque boxes—this prevents the model from learning to read. Each image was reshaped into a square (with distortion) and resized to 64×64 pixels. The images were randomly divided into 8,990 train (82.4%), 960 test (8.8%), and 960 validate (8.8%) partitions. There 1091 images for each species.

The machine learning model that you will train is a variant of reduced FireNet (Gupta et al. 2023) which is a smaller−scale SqueezeNet (Iandola et al. 2016). This tiny model (22,090 parameters) was selected for a combination of relatively rapid training/inference speed and its suitability for the North American *Acer* dataset.

Tasks

(1) Install Netron by typing `sudo snap install netron` in the terminal. Type your password when prompted.

(2) Create a TensorFlow environment to build and train the FireNet model.

   (a) Make a working project directory by typing `mkdir acer && cd acer` in the terminal.

   (b) Start a Docker build file by typing `echo 'FROM tensorflow/tensorflow:2.9.3' > Dockerfile` in the terminal.

   (c) Add a RUN statement to the build file by typing `echo 'RUN python3 -m pip install --upgrade pip && python3 -m pip install --upgrade setuptools && python3 -m pip install tensorflow-addons' >> Dockerfile` in the terminal.

   (d) Build the Docker image by typing `docker build -t 'tensorflow:2.9.3' .` in the terminal. This process may take several minutes to complete depending on your network speed.

   (e) Download the segmented data archive by typing `for k in {0..4}; do wget 'https://github.com/dpl10/phytoinformatics2023/raw/main/NA-Acer-data'$k'.tar'; done` in the terminal. Answer question (1).

   (f) Extract the archive by typing `tar -x -M -fNA-Acer-data{0..5}.tar` in the terminal. Three .tfr files should be output.

   (g) Download the model script by typing `wget https://raw.github.com/dpl10/phytoinformatics2023/main/firenet.py` in the terminal.

   (h) Download the training script by typing `wget https://raw.github.com/dpl10/phytoinformatics2023/main/trainImages.py` in the terminal.

   (i) Download the testing script by typing `wget https://raw.github.com/dpl10/phytoinformatics2023/main/testImages.py` in the terminal.

   (j) Download the fine−tune training script by typing `wget https://raw.github.com/dpl10/phytoinformatics2023/main/finetuneImages.py` in the terminal.

[1] *A. circinatum* Pursh, *A. glabrum* Torr., *A. grandidentatum* Nutt., *A. macrophyllum* Pursh, *A. negundo* L., *A. pensylvanicum* L., *A. rubrum* L., *A. saccharinum* L., *A. saccharum* Marshall, and *A. spicatum* Lam.

(k) Download a pretrained FireNet model by typing
`wget https://github.com/dpl10/phytoinformatics2023/raw/main/firenet-pretrain.h5`
in the terminal.

(l) Make the scripts executable by typing `chmod +x *.py` in the terminal.

(3) Start the Docker instance by typing `docker run -u $(id -u):$(id -g) --rm -it -v "$PWD:/tmp" -w /tmp tensorflow:2.9.3` in the terminal.'

(4) Construct a randomly initialized FireNet model by typing `./firenet.py -a 10 -f selu -i 64 -o firenet-i0.h5 -r 123456789` in the terminal. Answer question (2).

(5) View the untrained model in netron using the graphical interface or a web browser (https://netron.app). Answer question (3).

(6) Create a directory to save trained models to by typing `mkdir firstTrain` in the terminal.

(7) Train the model by typing `time ./trainImages.py -a 10 -b 64 -c -e 64 -f ce+clr+aw -i 64 -l 0.01 -m firenet-i0.h5 -o firstTrain -t NA-Acer-64-train.tfr -v NA-Acer-64-validation.tfr` in the terminal. Answer question (4).

(8) Evaluate the best model from the training by typing `./testImages.py -a 10 -c -i 64 -m $(ls -ltr firstTrain/*/best-model.h5 | awk '{print $9}' | tail -1) -t NA-Acer-64-test.tfr` in the terminal. Answer question (5).

(9) Create a directory to save trained models to by typing `mkdir secondTrain` in the terminal.

(10) Fine–tune the pretrained FireNet the model typing `time ./finetuneImages.py -a 10 -b 64 -c -e 64 -f ce+clr+aw -i 64 -l 0.01 -m firenet-pretrain.h5 -o secondTrain -t NA-Acer-64-train.tfr -v NA-Acer-64-validation.tfr` in the terminal. Answer question (6).

(11) Evaluate the best model from the fine–tuning by typing `./testImages.py -a 10 -c -i 64 -m $(ls -ltr secondTrain/*/best-model.h5 | awk '{print $9}' | tail -1) -t NA-Acer-64-test.tfr` in the terminal. Answer question (7).

(12) Close the Docker image by typing `exit` in the terminal.

Questions (https://forms.gle/5A1jaYFK6TKPfJQF8)

(1) For task (2)(e), explain what each part of the command does.

(2) For task (4), explain what each argument of the command does.

(3) For task (5):

(a) How many parameters does this FireNet model have?
(b) How many parameters are trainable?
(c) What is the size of the output layer?
(d) How large of an image will be processed by the model?

(4) For task (7):

(a) Explain what each part of the command does.
(b) How long (real time == wall clock time) did it take to train the model?
(c) What was the model's accuracy, AUPRC, and F_1 on the validation dataset?
(d) Did the model overfit during training?

(5) For task (8):

(a) What was the model's accuracy, AUPRC, and F_1 on the test dataset?
(b) Is this very different from the corresponding values on the validation dataset?
(c) Would the model benefit from additional epochs of training?

(6) For task (10):

(a) How many parameters does this FireNet model have?
(b) How many parameters are trainable?
(c) Compare and contrast the number of trainable parameters and their location in this model and in 'firenet-i0.h5'.
(d) How long (real time == wall clock time) did it take to fine–tune the model?
(e) How does this compare to the first training time?
(f) What was the model's accuracy, AUPRC, and F_1 on the validation dataset?
(g) Did the model overfit during training?

(7) For task (11):

(a) What was the model's accuracy, AUPRC, and F_1 on the test dataset?
(b) Which training worked better? Explain.

Literature cited

Gupta, K. D., , D. K. Sharma, S. Ahmed, H. Gupta, D. Gupta & C.-H. Hsu. 2023. A novel lightweight deep learning–based histopathological image classification model for IoMT. Neural Processing Letters 55: 205–228.

Iandola, F., S. Han, M. Moskewicz, K. Ashraf, W. Dally & K. Keutzer. 2016. SqueezeNet: AlexNet–level accuracy with 50x fewer parameters and $<$0.5mb model size. arXiv 1602.07360.

Due at the start of class May 2.