CPSC 585 - Artificial Neural Networks

Group Project 1, Spring 2023 - due March 25

In this project, you will work with a team to tackle a more complex character recognition problem using higher-resolution grayscale images of hand-written letters. This problem will require significantly more effort and will take longer to train.

# Project Teams

This project must be completed in a team of five or six. The instructor will assign teams for this project in Canvas.

See the following sections of the Canvas documentation for instructions on group submission:

* [How do I submit an assignment on behalf of a group?](https://community.canvaslms.com/t5/Student-Guide/How-do-I-submit-an-assignment-on-behalf-of-a-group/ta-p/294)

# Datasets

## EMNIST

The Extended MNIST or [EMNIST dataset](https://www.nist.gov/itl/products-and-services/emnist-dataset) expands on the [MNIST database](https://en.wikipedia.org/wiki/MNIST_database) commonly used as a benchmark, adding handwritten letters as well as additional samples of handwritten digits.

There are several “splits” of the data by various characteristics. We will be using the “EMNIST Letters” dataset, which contains values split into 27 classes, one unused (class 0) and one for each letter in the English alphabet.

*Note*: Some classes in this dataset can be challenging to recognize because each class contains images of both upper- and lower-case letters. For example, while ‘C’ and ‘c’ are very similar in appearance, ‘A’ and ‘a’ are quite different.

The file [emnist\_letter.npz](https://drive.google.com/file/d/1m6FN1CrVDlQSvrbcLQmI-J81w5kDxX8P/view?usp=sharing) contains EMNIST Letters in a format that can be opened with the [numpy.load()](https://numpy.org/doc/stable/reference/generated/numpy.load.html) method. The data contains six arrays: 'train\_images', 'train\_labels', 'validate\_images', 'validate\_labels', 'test\_images', and 'test\_labels'. The values have been adjusted from the original EMNIST dataset in order to match the MNIST examples included with Keras:

* The images have been transposed and scaled to floating point.
* The labels have been one-hot encoded.

## Binary AlphaDigits

The [Binary Alphadigits](https://cs.nyu.edu/~roweis/data.html) dataset contains another set of handwritten letters and digits, in a different image size, in bitmap format.

The file [binaryalphadigits.npz](https://drive.google.com/file/d/1C1S0lSR140MRXil035PTd5KbfTQd2hwT/view?usp=sharing) contains the letters from this dataset, in a format that can be opened with the [numpy.load()](https://numpy.org/doc/stable/reference/generated/numpy.load.html) method. The data contains two arrays: 'images' and 'labels'. The values have been adjusted from the original Binary Alphadigits dataset in order to match the EMNIST Letters:

* The images of digits have been omitted.
* The labels are in the same format as EMNIST Letters, including the unused class 0.

Note, however, that the resolution of the images is different in this dataset: 20×16 rather than 28×28.

# Tasks

## Part 1 - Warm-Up

1. Open [this notebook](https://colab.research.google.com/github/fchollet/deep-learning-with-python-notebooks/blob/master/chapter02_mathematical-building-blocks.ipynb#scrollTo=6yxiq707ksqH) by Francois Chollet, which creates a simple Multilayer Perceptron as described in Section 2.1 of *Deep Learning with Python, Second Edition*. (Recall that this book is available from the library’s O’Reilly database.)

Chollet’s example uses the simpler MNIST dataset, which includes only handwritten digits. That dataset is [included with Keras](https://www.tensorflow.org/api_docs/python/tf/keras/datasets/mnist).

Run the model from this notebook. What accuracy does it achieve for MNIST?

1. Load the EMNIST Letters dataset, and use [plt.imshow()](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html) to verify that the image data has been loaded correctly and that the corresponding labels are correct.
2. Apply the network architecture from Chollet’s MNIST notebook to the EMNIST Letters data. (You will need to modify the number of outputs, but should leave the dense layer intact.)

*Note*: You are welcome to use PyTorch to implement the same architecture; Keras is required only to run step (1).

What accuracy do you achieve? How does this compare with the accuracy for MNIST?

1. The Keras examples include a [Simple MNIST convnet](https://keras.io/examples/vision/mnist_convnet/). Note the accuracy obtained by that code compared to the previous example from Chollet.

Apply the same architecture to the EMNIST Letters data. (Again, you are welcome to implement an equivalent architecture in PyTorch instead). What accuracy do you achieve? How does this compare with the accuracy for the MNIST? How does it compare with the accuracy for EMNIST that you saw with a Dense network in step (3)?

## Part 2 - Main Event

1. You should have found that while the EMNIST Letters are harder to learn than the MNIST digits, switching to a different network architecture led to a significant increase in model performance.

You may have noticed, however, that the training process was slower. This means that experiments take longer, and mistakes can be costly. While plotting a learning curve when training has finished can help diagnose problems, ideally you will want to see updates during the training process.

In order to avoid dead-ends while adjusting and tuning your model, TensorFlow includes the [TensorBoard](https://www.tensorflow.org/tensorboard/get_started) tool and the [TensorBoard notebook extension](https://www.tensorflow.org/tensorboard/tensorboard_in_notebooks) for this purpose. While the examples show Keras models, [PyTorch supports TensorBoard as well](https://pytorch.org/tutorials/recipes/recipes/tensorboard_with_pytorch.html).

Add TensorBoard support to the CNN model you run in Part 1, and add the TensorBoard extension to your notebook to visualize the training process.

*Note*: if you get a 403 error when trying to use TensorBoard in Google Colab, you may need to [enable third-party cookies](https://stackoverflow.com/a/65221220).

1. Now that you have a baseline convolutional network for comparison, begin experimenting with alternative architectures (e.g. adding additional filters to learn features and additional hidden layers to learn combinations of features) and with adjusting hyperparameters.

Your team’s goal is to obtain as high an accuracy as possible on the validation set.

Use what you’ve learned in Chapters 2 through 5 of the textbook to obtain the highest accuracy you can, including:

* Weight initialization
* Choice of activation function
* Choice of optimizer
* Batch normalization
* Data augmentation
* Regularization
* Dropout
* Early Stopping
* Pooling

(You will notice that some of these techniques were already in use in the Keras [Simple MNIST convnet](https://keras.io/examples/vision/mnist_convnet/) example.)

1. When you are satisfied with your model's performance, save your model and evaluate the results on the test set.

## Part 3 - Transfer Learning

1. The process of transfer learning can be used to apply an existing model to a new dataset. See [Transfer learning & fine-tuning](https://keras.io/guides/transfer_learning/) in the Keras Developer Guide or the [Transfer Learning for Computer Vision Tutorial](https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html) in the PyTorch Tutorials.

The images in the Binary Alphadigits dataset are a different size from those in EMNIST Letters. Use a function like [tf.image.resize\_with\_pad()](https://www.tensorflow.org/api_docs/python/tf/image/resize_with_pad), [PIL.ImageOps.pad()](https://pillow.readthedocs.io/en/stable/reference/ImageOps.html#PIL.ImageOps.pad), or the PyTorch [torchvision.transforms.Resize](https://pytorch.org/vision/stable/generated/torchvision.transforms.Resize.html#torchvision.transforms.Resize) class to resize them into the right format for the network you trained in Part 2.

1. Is the model you trained in Part 2 capable of recognizing letters from this new dataset?
2. Can you improve the performance on this dataset by adding additional trainable layers and fine-tuning the network?
3. Compare the performance of the model you built in step (3) with the performance of a brand-new model trained only on the Binary AlphaDigits dataset.

*Note*: the dataset is so small that this may be a valid use-case for cross-validation.

What do you conclude about the value of reusing a pre-trained model?

## Platform

Perform the tasks above and document their results using a notebook on [Google Colab](https://colab.research.google.com/) with your @csu.fullerton.edu account. If you are not familiar with Google Colab or Jupyter Notebooks, the [Welcome To Colaboratory](https://colab.research.google.com/) notebook should help you get started. Note, in particular, the section [Using Accelerated Hardware](https://colab.research.google.com/#using-accelerated-hardware).

While you may choose to work locally, especially if you have access to a physical machine with a GPU, your project submission must be uploaded to Google Drive and run successfully in Colab.

## Libraries

In addition to [TensorBoard](https://www.tensorflow.org/tensorboard), image processing libraries like [Pillow](https://pillow.readthedocs.io/en/stable/) and the libraries from the previous project, you are welcome to use other libraries such as [fast.ai](https://docs.fast.ai/) or [PyTorch Lightning](https://pytorch-lightning.readthedocs.io/) if you find them helpful.

# Documenting your results

Notebooks allow you to create documents mixing text, equations, code, and visualizations. Your project should make good use of these features. For example:

* Identify each task to be performed, documenting any decisions made.
* Include both the code to perform each task and its output. Where appropriate, tasks should be broken up into separate blocks, with the results shown for each.
* Include written analysis of results along with code output and visualizations.

In short, a reader unfamiliar with the project should be able to read your notebook and understand what you did and what results you obtained.

# Submission

From inside your Google Colab notebook, use the **Share** button at the top-right of the toolbar to [share your notebook](https://colab.research.google.com/notebooks/basic_features_overview.ipynb#scrollTo=aro-UJgUQSH1) with the professor:

1. Make certain that you are logged into Colaboratory with your @csu.fullerton.edu email address.
2. Add the professor’s @fullerton.edu email address as a **Viewer**.
3. Leave **General access** set to ***Restricted***, rather than setting it to ***Cal State Fullerton*** or ***Anyone with the link***.
4. Use the **Copy Link** button to copy the link to the clipboard.
5. [Submit the link](https://community.canvaslms.com/t5/Student-Guide/How-do-I-enter-a-URL-as-an-assignment-submission/ta-p/286) you copied via Canvas by the deadline.

## Grading

The project will be evaluated on the following five-point scale, inspired by the [general rubric](https://cs533.ekstrandom.net/f21/assignments/#general-rubric) used by Professor Michael Ekstrand at Boise State University:

**Exemplary (5 points)**

The project is a success. All requirements met. The quality of the work is high.

**Basically Correct (4 points)**

The project is an overall success, but some requirements are not met completely, or the quality of the work is inconsistent.

**Solid Start (3 points)**

The project is mostly finished, but some requirements are missing, or the quality of the work does not yet meet professional standards.

**Serious Issues (2 points)**

The project has fundamental issues in its implementation or quality.

**Did Something (1 point)**

The project was started but has not been completed enough to assess its quality fairly or is on the wrong track.

**Did Nothing (0 points)**

The project was not submitted, contained work belonging to someone else, or was of such low quality that there is nothing to assess.