

Final Term Reflection

This document contains my responses and analysis, all referenced files are listed in ./ (current directory)

PyData: I can use the tools of the PyData stack to understand, interpret, and visualize datasets, including making arguments about its underlying distributions: overarching understanding of ml, and the ability to use the tools necessary throughout the process.

The past few weeks, notably starting with my MNIST model that I created for the Oral Exam, have been comprised of an extremely deep dive into the tools used for machine learning. This term I was excited by the prospects of Optical Character Recognition – and particularly its application onto Chinese characters. Training a model to recognize digits (MNIST) or English letters (EMNIST, Kaggle) is relatively straightforward as the number of outputs is low. In comparison, a Chinese character recognition algorithm must have at least 4,000 potential outputs (depending on the character encoding used). As such, every aspect of an MNIST or EMNIST model must be scaled to match, both in data resolution, datum count, CNN nodes, and processing power.

Mechanically, I have a strong grasp of the libraries and functions necessary for the creation of an algorithm and model. I feel comfortable with the manipulation of data using Numpy and Matplotlib (see data stories section) and adept in my understanding of the multi-dimensional analysis necessary for the manipulation of tensors (see data modeling). I am competent in TensorFlow, and by this point in the term, have completed two fully-functioning models with the framework (see attached model + processing.py, keras.py).

Item	Score
Organization	3
Volume of Work	3
Analysis/Documentation	3
Progress	3
Total	12

Data Stories: I can tell stories with data, both by discussing my process in

shaping/manipulating/modeling it and the choices made to do so, and through making arguments about what my findings say about the world.

The ability to tell stories with data has remained core to my work throughout the term. By a large margin, the most difficult part of the last few weeks has been working with data and its various presentations across the field of machines learning. While some data providers have their data stored in commonly used or self-explainable file types (see csv, json), many systems will create proprietary file types and code dependencies depending on what format works best with their data. Coupled with the lack of clear documentation / tutorial resources in less

commonly used collections, it becomes imperative to fully understand how to interpret, organize, and best process your data for model preparation.

Data provided by CASIA-HWDB (CASIA handwritten database) isn't contained within columns or rows. Instead, the files are simply a coenacted sequence of bytes that depict the values at each point in the set. The data cannot be decoded for viewing in a text editor, as each segment of the sequence uses a different codec (UTF-8, ASCII, and GBK). The order is described by this table on the provider's website:

Table 2. Format of offline isolated character data file (*.gnt)

Item	Type	Length	Instance	Comment
Sample size	unsigned int	4B		Number of bytes for one sample (byte count to next sample)
Tag code (GB)	char	2B	"阿"=0xb0a1 Stored as 0xa1b0	
Width	unsigned short	2B		Number of pixels in a row
Height	unsigned short	2B		Number of rows
Bitmap	unsigned char	Width*Height bytes		Stored row by row

The process of understanding each of these parameters could not be completed using the standard path of analysis through tools such as Matplotlib or Seaborn (scatterplots, histograms). Instead, I needed to create a python function that could iterate over and process each byte separately. To accomplish this, I created this function:

```

def byte_decoder(directory):
    x, y = [], [] # initialization of data storage: x = character bitmaps, y = response feature (int/char)
    characters = [] # character array used for indexing responses

    for filename in glob.iglob(f'{directory}/*'): # iterates over all files in {directory}

        cx, cy = [], [] # current bitmap/character set, reset for each file
        with open(filename, 'rb') as data: # with open() processes and closes each file (by byte: rb)
            print(filename) # status check for processing, as it takes around 10 minutes for all of the data

            bytes_listed = [] # all bytes per file
            offset = 0 # position in bytes array
            while (byte := data.read(1)):
                bytes_listed.append(byte) # breaking file data into array

            while offset < len(bytes_listed):
                # iterating over each byte, while loop is preferred here for simultaneous processing
                current_sample_size = bytes_listed[offset] + bytes_listed[offset + 1] + bytes_listed[offset + 2] + \
                    bytes_listed[offset + 3]
                # currently unused data point measuring distance to next item, used to replace missing data
                offset += 4

                current_character = 0 # reset char value
                current_character, characters = character_sort(byte_to_num((bytes_listed[offset]
                    + bytes_listed[offset + 1])), characters)
                cy.append(current_character) # adds corrected character to current storage
                offset += 2

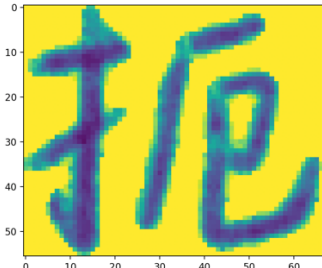
                current_bitmap_size = 0 # initializes bitmap parameters
                current_bitmap_height = 0
                current_bitmap_width = 0
                current_bitmap_height = byte_to_num((bytes_listed[offset] + bytes_listed[offset + 1]))
                current_bitmap_width = byte_to_num((bytes_listed[offset + 2] + bytes_listed[offset + 3]))
                current_bitmap_size = current_bitmap_width * current_bitmap_height
                offset += 4

                current_bitmap = [] # initializes bitmap
                current_bitmap_size_tracker = 0
                while current_bitmap_size_tracker < current_bitmap_size: # iterates over every byte in bitmap
                    current_bitmap.append(byte_to_num(bytes_listed[offset + current_bitmap_size_tracker]))
                    current_bitmap_size_tracker += 1
                cx.append(dimensionality_buffer(current_bitmap, current_bitmap_width, current_bitmap_height))
                # converts list -> numpy array and pads/crops values
                offset += current_bitmap_size
                current_bitmap_size = 0

        for item in cx:
            x.append(item) # combining current bitmap with total array
        for item in cy:

```

The function parses the file byte-by-byte (as described in the comments) and divides the data into two arrays: the bitmap feature, and the corresponding char value response. This can be confirmed by the graphing and decoding of the two items at any constant index:

Bitmap represented by Matplotlib	Character response (decoded to GBK)
	扼

Finally, each image had to be prepped via the padding and cropping of data to enforce consistent sizing throughout the process:

```
def dimensionality_buffer(byte_list, width, height):
    byte_array = numpy.array(byte_list).reshape((width, height))
    # converts byte list to 2 dimensional array
    shape = numpy.shape(byte_array)
    padded_array = numpy.full((500,500), 255)
    # creates padding array of size (500,500) with blank values 255
    padded_array[:shape[0],:shape[1]] = byte_array
    # superimposes bitmap (bytearray) data onto blank padding set
    return padded_array[0:100, 0:100]
    # crops final output from (500,500) to (100,100) based on average dimension values
```

Overall, I feel very accomplished in my progress with understanding the 'language' of data. I recommend looking through my entire processing.py file attached, as it outlines a more easily understandable walkthrough for my code. (In comparison to my midterm reflection, I have shifted my understanding of this objective slightly as it pertains to my work, for the 'telling of stories with data' refer more to my morality/ethicality reflection, as I feel like that is a better representation of the division)

Item	Score
Organization	3
Volume of Work	3
Analysis/Documentation	3
Progress	3
Total	12

Data Modeling: I can implement and describe the use of all aspects of the data

modeling process: the processing, understanding, and manipulation of data, both before, during, and after the ml modeling process.

Through my independent work this term, I have made immense progress in understanding each part of the modeling process. In comparison to my midterm reflection, at which point the extent of my work with real models was surface level and little more than an understanding of the term, I have been able to focus on and ‘master’ (very rough mastery, of course, considering the few weeks we had this term) the parts of the field that I find most interesting. I am comfortable in my proficiency in Convolutional Neural Networks, their hyper-parameters, and the changes in abstraction as tensors are processed layer-by-layer.

```
def define_model(): # layer definitions
    model = Sequential() # layers are sequential, each tensor is passed through all layers in order
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(100, 100, 1)))
        # breaks bitmap image into (3,3) kernels with 64 filters, images of shape (100,100)
    model.add(MaxPooling2D((2, 2))) # pools and converts kernels from (3,3) to (2,2)
    model.add(Flatten()) # flattens data to 1 dimension
    model.add(Dense(5000, activation='relu', kernel_initializer='he_uniform'))
        # densely-connected layer with 5000 nodes
    model.add(Dense(4037, activation='softmax')) # response layer, using softmax: determining a single output value
    opt = SGD(learning_rate=0.0001, momentum=0.9)
        # optimization learning parameters, best fit for this set based on my testing
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

This excerpt is from my modeling on CASIA-HWDB (Chinese character OCR). The program is a CNN and comprised of layers that successfully abstract my input data into an output. It is configured to receive bitmaps, at (100,100) resolution and map each path to the 4037-neuron output layer, with each output corresponding to a single Chinese character or symbol.

```
def evaluate_model(dataX, dataY, n_folds=5): # builds model
    kfold = KFold(n_folds, shuffle=True, random_state=1) # divides data into 5 shuffled subsets for training
    for train_ix, test_ix in kfold.split(dataX): # iterates over each k-fold validation split
        model = define_model() # layer initialization
        trainX, trainY = dataX[train_ix], dataY[train_ix] # training data for split
        testX, testY = dataX[test_ix], dataY[test_ix] # testing data for split
        model.fit(trainX, trainY, epochs=10, batch_size=32, validation_data=(testX, testY), verbose=1)
            # fits model for 10 epochs, low batch size of 32 to increase fit tightness on provided data
        _, acc = model.evaluate(testX, testY, verbose=1) # returns accuracy, verbose = 1 displays running processes
        print('> %.3f' % (acc * 100.0)) # prints per fold completion (5 total)
```

Above is the code I use to initialize my algorithm. I utilize K-Fold validation to divide and shuffle the data across 5 folds (trials) and build the model using the divided data from each. I found that a batch size of 32 was the optimal value, minimizing my loss to an asymptote of 8 (although this is relative, of course) and increasing my accuracy to ~3-4%.

To see my full documented work for the creation of my algorithm, I recommend reading the entirety of model.py. Additionally, if you would like, check the commented testing code from the bottom of my model and processing documents to see some of the strings I put together to test my model. To see a complete and accurate model (averaging between 98-99% accuracy) I recommend checking my keras.py document.

Item	Score
------	-------

Organization	3
Volume of Work	3
Analysis/Documentation	3
Progress	3
Total	12

Ethical Reasoning: I can use ethical reasoning to empower my data decisions, ensuring that the technical work that I do promotes equity and justice.

Ethical reasoning has been the final core component to my work this term. Throughout the past weeks, I have spent hours reflecting on the morality of my decisions in machine learning. Much of this work has been interpersonal, through conversations and extra-curricular projects. But I have made sure to record much of my thinking as they reach significance in my personal projects. Attached below I have the article I wrote in correspondence with my OCR work this term, and an evaluation of the potential impacts it could have if it were scaled and deployed to users.

```

# CASIA-HwDB

# USER EXPERIENCE AND ETHICAL ANALYSIS

# In comparison with many of the subsets of machine learning, Optical Character Recognition has little
# impact onto the wellbeing of its users. It does not determine ones societal worth, diagnose life-
# threatening disease, or threaten labor work. Still, there are a few key areas where its impact can be
# negative -> depending on the accuracy, training, and application of the model and its decisions.

# To describe the potential use cases and harm that could be caused as a result of them, I will outline
# my main idea for the application of my model.
# ----- #
# I would like to use OCR to develop a language learning application. The app
# (built in SwiftUI) would integrate with TensorFlow using CoreML. The app would
# allow users to practice Chinese by drawing characters, at which point the app
# abstracts the drawing into a bitmap and returns the most likely character that
# has been drawn
# ----- #

# Communication is essential, and any application that influences ones ability to communicate can have
# negative effects. If the model has too low an accuracy, then it has the potential to mislead its users
# and incorrectly identify characters. If the knowledge that the user gained from the application were to
# be used in a serious setting, such as a court case, it could cause life changing error. It is nearly
# impossible to create a classification algorithm with 100% accuracy. The bulk of the ethical dilemma
# occurs in consideration of the line that must be drawn between accuracy and inaccuracy. With a large
# enough pool of users, detrimental error is inevitable. Is 98% accuracy sufficient? 99%? 99.5%? The most
# ethical route of action would likely be to tell the user that the model's results should not be trusted;
# however, at that point, there isn't really a point in using the model at all. As an auxiliary point, the
# issue could be diminished by making the model return a prediction of the most likely values, as opposed
# to a single character - increasing its chance of guessing the correct value (provided that the model has
# a low enough loss, of course)

# Additionally, there is the issue of representation within the training data. Chinese is heavily influenced
# by dialect, and as such, there are variations in characters across regions. CASIA-HwDB, the set used for
# this analysis, was compiled by 320 - 420 writers (depending on the percentage of data accessed) from Beijing.
# The data does not fully encompass the specific styles and patterns of more rural regions.

```

Additionally, I spent time taking note of articles that I found interesting – writing short reflections about their significance and collecting my responses in a folder (see attached files). While I feel satisfied with the depth that I reached in my understanding of the ethics of ML this term, especially in consideration of where I started, I wish that I had spent some more time with the material. In consideration of this class as a 600-level course, I could have been better about balancing my time with AI’s ethics. This is why I am removing score for the “volume of work” section of this objective.

Item	Score
Organization	3
Volume of Work	2
Analysis/Documentation	3

Progress	3
Total	11

Additional Questions: Suppose you could travel back in time to the start of the term to give your 10-weeks-ago-self some advice for success in this course. What advice would you give yourself? How do you feel you did at adhering to this advice? Anything else you'd like me to know?

If I could return to the start of this term, I would tell myself to be more confident in trying new ideas – inside and outside of our code – and to embrace mistakes as they happen. I can clearly remember you describing the feeling of being ‘utterly lost’ during one of our first classes, and its importance to any time that we may spend in computer science. I wish that I had been more ready to embrace this, making more broken code than waiting for my understanding to catch up. I have found that one of the best ways for me to improve has been by making mistakes, which I have spent the last few weeks doing, and I wish that I had gained the confidence to fail sooner.

I have really enjoyed taking CSC630 this term, and I feel inspired to keep working in ML. I want to continue improving my Chinese character recognition model, and I am excited about the prospect of its integration into an iOS app that I and others could use to study the language – although that may be a while off, considering the gravity I now hold to the project derived from my ethical reflection.

Item	Score
Total	47