Capstone 2, Milestone Report 1

Problem Statement

In this project, I will tackle the problem of character recognition in natural scenes - that is, images obtained through photographs of objects in the real world such as street signs, business signs, and other sources of text, rather than images of computer fonts or similarly normalized text. These “natural” images display a far greater degree of variability in text styles than computer fonts, so they pose a more significant challenge in the realm of computer vision; classical OCR methods do not perform nearly as well on natural images as they do on computer-generated text.

The applications of character recognition in these contexts are very broad. Some of the tasks enabled by this technology are automatic driving, testing the robustness of CAPTCHA anti-bot security, and developing tools to assist the blind or visually impaired.

The dataset I will use for this project is the Chars74K dataset, which was composed by Teo de Campos, et al, alongside a publication describing the team’s methods and results. The full dataset includes sets of both English alphanumeric characters and Kannada characters, as well as a number of handwritten and computer font characters. I intend to focus on the English alphanumeric characters set, which contains 7705 “good” images. An additional 4798 “bad” images are included in a separate folder, but these were not included in the published study due to their excessive occlusion, noise, or low resolution, so I likewise will use only the “good” part of the set.

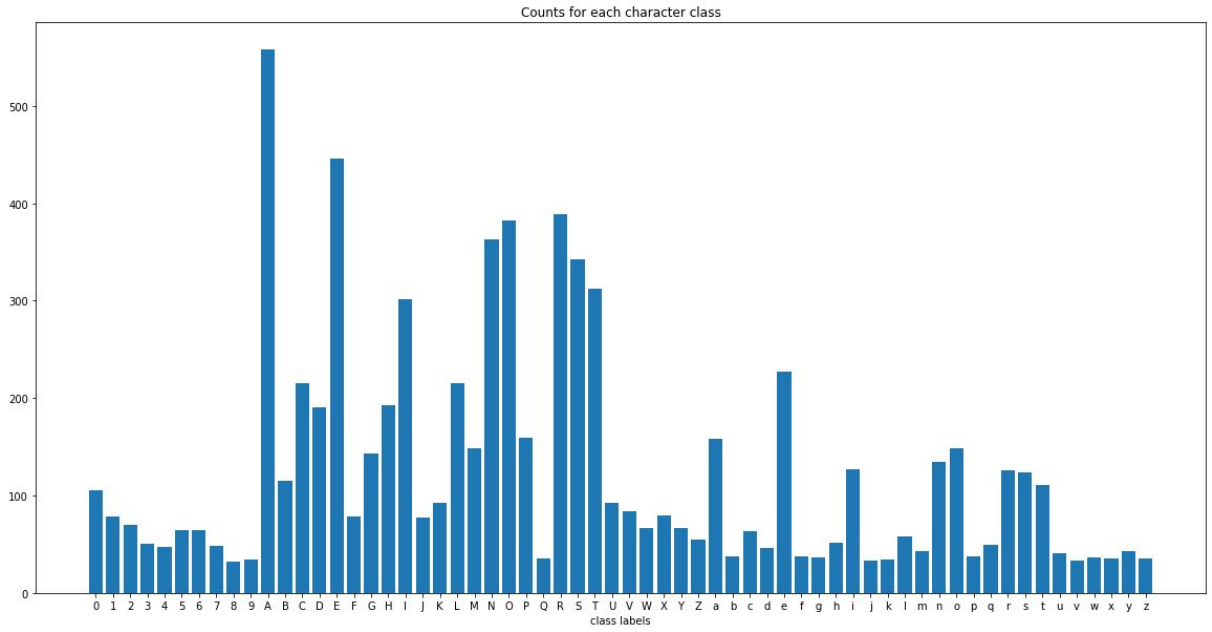
I intend to use a convolutional neural network (CNN) for this project. CNNs are a powerful tool for image classification problems, so they are well suited to the project I have laid out here. Upon completion of the project, I will have a final report, a slide deck, and project code to deliver.

Dataset Description

As I mentioned above, the segment of the Chars74K dataset that I intend to use contains 7705 images. These images may be divided into 62 classes: 10 for each digit including zero, 26 for each uppercase character, and 26 for each lowercase character. Obtaining the dataset involved only downloading it from the Chars74K web page. The dataset is mostly well-maintained; however, 5 images, all from the lowercase-q subset, had to be removed from the sample set due to file type incompatibility. Since only a very small number of images were removed, I do not expect this to meaningfully impact the model's results.

Cleaning and Exploration

After obtaining the data, I performed a number of preprocessing and data exploration steps. First, I counted the number of samples in each class, which I have plotted below.

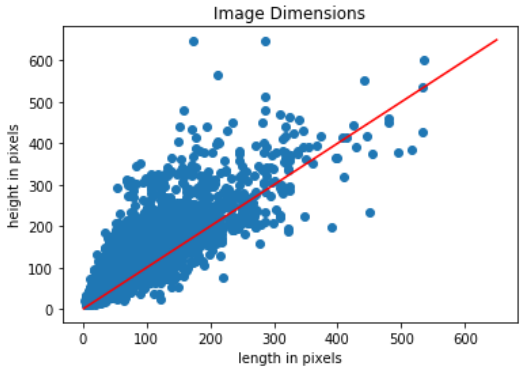


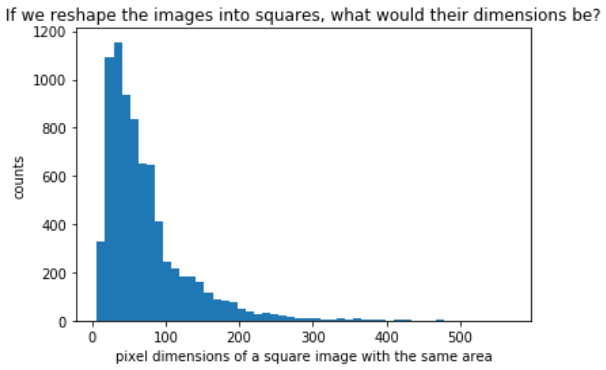
It is immediately apparent that there is significant class imbalance in this dataset. There are a few ways I could choose to deal with this. Teo de Campos recommends testing and training on only 15 samples from each class, though I have seen others use data augmentation or compute class weights instead. I initially considered it important to be able to draw direct comparisons between my model and the original Chars74K paper,

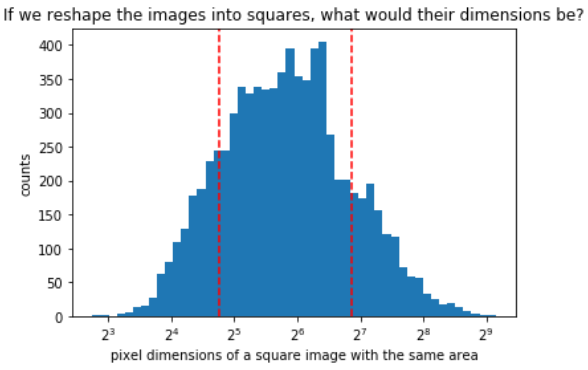
but it seems that enough work has been done using this dataset during its lifetime that this is not necessarily the case. With that in mind, I opt to compute class weights since this is the simplest technique to implement.

Some other statistics to note about the class distribution are its maximum value, 558 for capital A; its minimum value, 32 for number 8; the corresponding range, 526; and the mean and median values, 124.19 and 78. For those classes with a low number of samples, the fraction used to train the model approaches 50%, though this fraction is significantly lower for classes that have more samples.

In order to feed the images to the model, they will all need to have the same dimensions, and the original images have a wide range of dimensions. In order to determine appropriate dimensions for the images, I create a scatter plot of the length and width of the images and two histograms derived from the images' total areas. For the histograms, the actual values plotted are the square roots of the areas, which tells me what length in pixels to use; the first histogram is plotted on a linear scale and the second on a logarithmic scale.







The scatter plot offers some clues that a majority of the data are clustered toward the smaller dimensions, which is confirmed by the first histogram. Although most of the

images appear to be taller than they are wide, I settle on using square dimensions for the sake of model simplicity since this seems to be a reasonable approximation of the data. Furthermore, data augmentation used to increase the sample size of image sets often involves stretching and resizing images in order to create distinct, yet still recognizable, variations of a class, so I believe that any transformations induced by this uniform resizing strategy are more likely than not to yield better predictions.

One technique that caught my interest when researching convolutional neural networks was progressive resizing. I will save a more technical explanation for when I get to actually building the model; for now, the thing to know is that it will involve successively doubling the images' dimensions. With this in mind, I select a range of pixel dimensions that contains the largest number of images, which, due to the logarithmic plot's approximately normal distribution, happens to occur at the central peak of that plot. Based on this data, I decided to use pixel dimensions of 32x32, 64x64, and 128x128.