English Alphabet Prediction

Daniel Gunleiksrud, Sunniva Lothe, Sayna Ganjei, Vladimir Civilgin

https://github.com/591291-hvl/Letter-Classification

DESCRIBE THE PROBLEM

SCOPE

The dataset that we are working on is the EMINST dataset. Considering the dataset is well known for character recognition, it is often used to benchmark image classification models; there are numerous models getting accuracy above 90%. Others have solved it with both the upper and lower case characters. We only used uppercase characters in our model, so there is potential for expansion.

Classifying characters is not a simple task without machine learning. It would be ineffective, and time consuming to write such an algorithm, due to unclear handwritten characters. The characters can also drastically deviate from each other depending on who wrote them. Computer fonts however, are less complicated to write an algorithm for since most modern fonts are designed to have distinct characters.

Before getting started with building the model, we have to frame the problem and think about the goal after building the model, the business aspect of it. The business objective helps to make major decisions and plan for the future. We need to ask ourselves “How do we expect to use and benefit from the model?”.

The algorithms that will be used are essential for being able to predict on a handwritten character. Gaussian Blur, ROI Extraction, Centered Frame, and bi-cubic interpolation will not take a long time to process on a single 128x128 image. For predicting the characters, we use a neural network model, because it is the most suitable for image recognition. The average processing time of the algorithms is under a second. This is beneficial for incorporating our solution into bigger systems.

Our solution has two simple components, preprocessing and predicting. When incorporating it to more extensive “pipelines” or systems, little to no changes will be necessary. The image converter may need a slight change to handle images of any size.

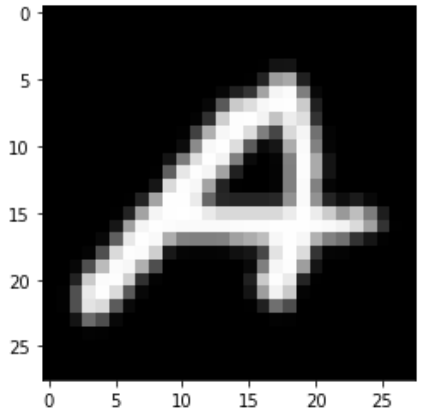
METRICS

To measure the performance of the solution we will use accuracy and confusion matrix. Accuracy determines the chance that the model will predict the correct character for a hand drawn image. This is a good evaluation to use , considering character classes are roughly balanced. The confusion matrix inspects if there are any characters that the model repeatedly struggles with.

When predicting on handwritten characters, it is essential that the model gives out a correct, or at least a close enough character, otherwise it is considered a failed model. If the model fails, it cannot be used in our system, or be incorporated in bigger systems.

DATA

For this project we will be using EMNIST dataset. This dataset contains handwritten images of characters and numbers along with correct labels. There are 62 different labels in this dataset, 26 labels for lowercase, 26 labels for uppercase and 10 for numbers. The images are 28 pixels by 28 pixels represented as a matrix, where each value can range from 0 to 255.



Above is an example of how an image looks. We can see that a value of 255 will give a full white pixel, whilst a lower value will give a darker pixel and 0 will give a completely black pixel.

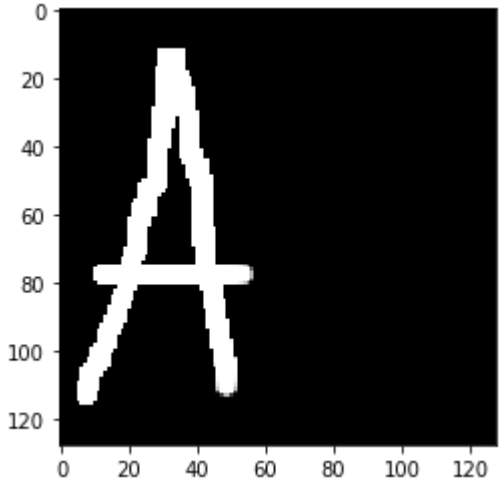
Currently only uppercase characters are used, as the accuracy of the prediction drastically decreases when including numbers and lowercase characters. By only including uppercase characters, 188 958 out of 697 932 images need to be processed to train the machine learning model, which makes it more effective and less time consuming. On the other hand, without numbers at lowercase characters, the model will have limited usage.

All the images in the dataset are 28x28, we would need the images we are going to predict on to also be 28x28. The plan is to write on a canvas html element, and since writing on a 28x28 pixel canvas is very small we have two obvious potential solutions:

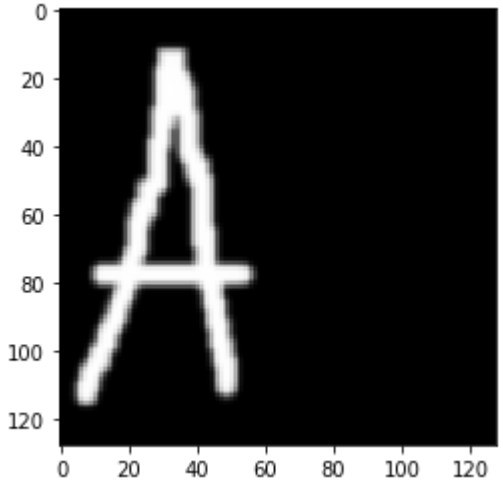
1. Upscale the canvas element:
   By doing this we circumvent having to do any form of image preprocessing. This results in each prediction requiring less processing power, thus faster user experience.
2. Have a larger canvas element but preprocess the image to 28x28 before predicting on it:
   By doing this we will spend more time reducing the image size so we don't lose valuable information. In return we have a more flexible method for predicting images. So we might not be limited to any canvas size, thus more friendly user experience.

For this project we chose the latter, as we want the option to be more flexible and have potential for expanding our project. In https://arxiv.org/pdf/1702.05373v1.pdf there is described a conversion process used to convert the NIST dataset to EMNIST. We can use the same method to reduce our images to 28x28. By using this method we should ensure that images of any size can be used. Although the NIST dataset is using binary values, this should be fine since there is little variance between 255 and 0.

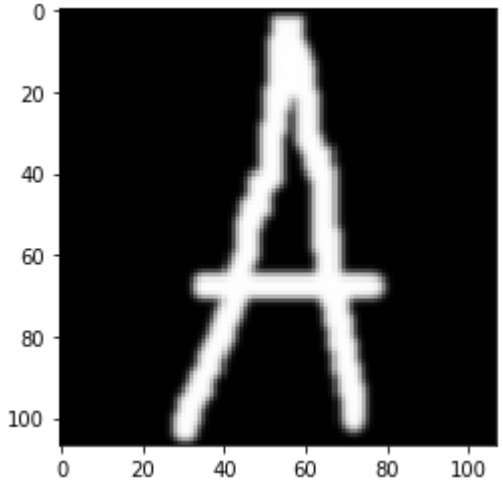
It is important that we do these steps as we want the images to resemble the training data as much as possible.
Step 0: Original image



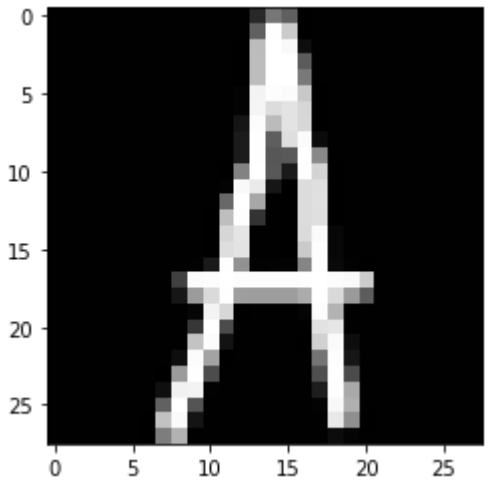
Step 1: We apply a gaussian filter. This is to blur the image a little, creating a smooth value transition from 255 to 0.



Step 2: ROI extraction. Remove padding around the values in the image. This makes it so that non-zero values are along the edge.
(Step 2 and 3 are done in one go, so can't get example image)
Step 3: Center frame. Add padding so we get a square image.



Step 4: Resized and resampled. Resized with using bi-cubic interpolation

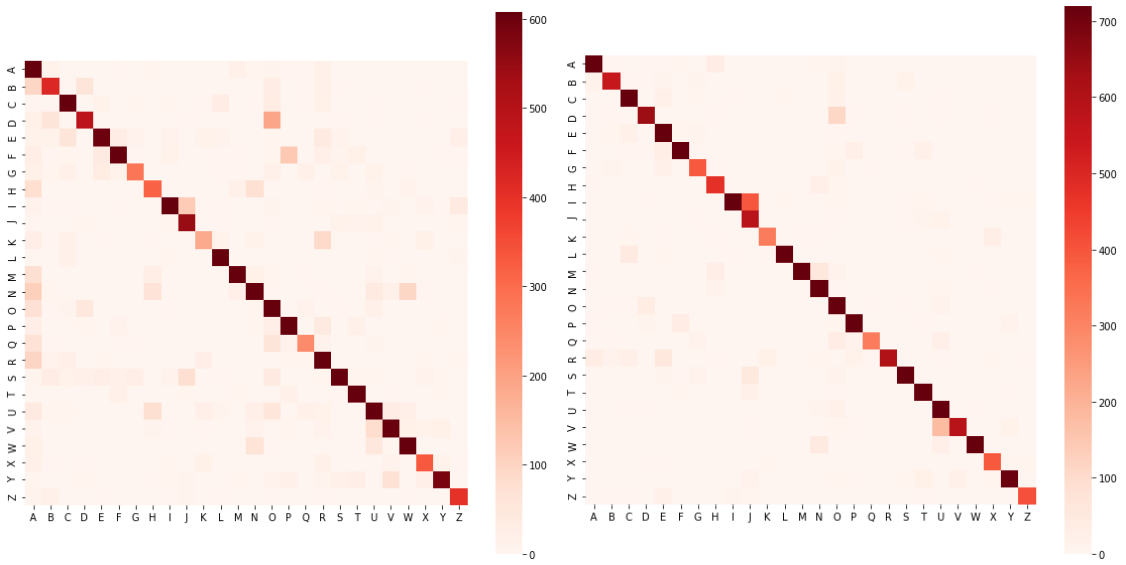


The result we end up with is an image that resembles the training data we used to train the model.

MODELING

We decided to use tf.keras.Sequential as our machine learning model. This is a deep learning model that allows us to stack neural net layers. By having input layers as how many pixels we have, and output nodes as number of classes(uppercase characters) we can use this as a baseline model. After we have our baseline model we can add further layers to hopefully improve the accuracy of our model.

Baseline model with layers 784x26 gives an accuracy of 0.8622. We also tested for layers of size 784x784x128x26 and this gives an accuracy of 0.9461.



The first image is a heatmap for the baseline model, and second is the improved model we are using. As you can see there has been a notable improvement where it used to predict A but was not A. The inaccuracy between i and j, and u and v still remains.

DEPLOYMENT

The model will be deployed on a website where the user is able to draw an uppercase character in a canvas element. The image written in the canvas element will be base64 encoded and sent with a post request. There it's decoded and preprocessed and predicted on before being sent back. All this happens in under a second before displaying the result to the user.

REFERENCES

Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. Retrieved from http://arxiv.org/abs/1702.05373