Handwriting Recognition

Berdan Akyürek	21600904
Asım Güneş Üstünalp	21602271
Turan Mert Duran	21601418
Maryam Shahid	21801344
Mannan Abdul	21801066

Work Done

- Preprocessing
- Using a CRNN approach
- Creating model by using kNN classifier

Introduction

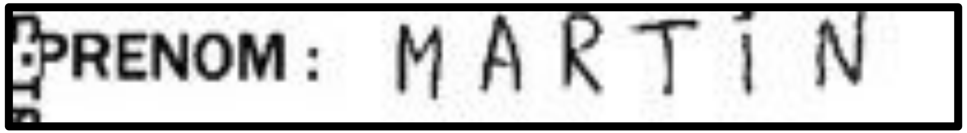
- Our lives are becoming increasingly intertwined with the digital world nowadays
- A need to be able to share information across the physical and digital aspects of our lives.
- Exploring different Machine Learning approaches to handwriting recognition.

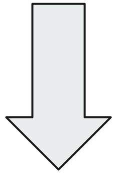
Dataset

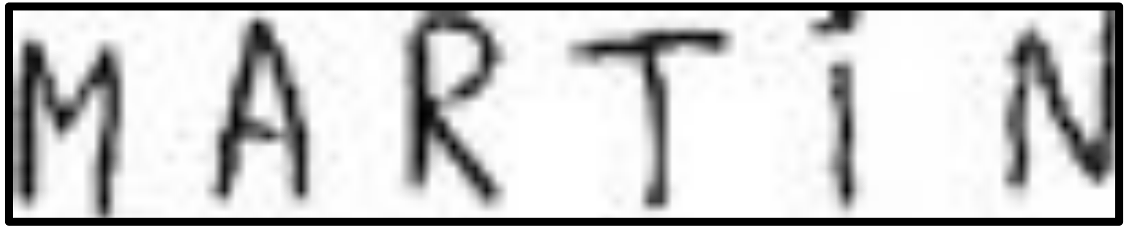
Our Dataset [2] contains:

- 330,961 training images
- 41,370 validation images
- 41,370 test images
- 3 CSV files that contain the URL of the images in the dataset along with their labels

Preprocessing (Cropping to Content)

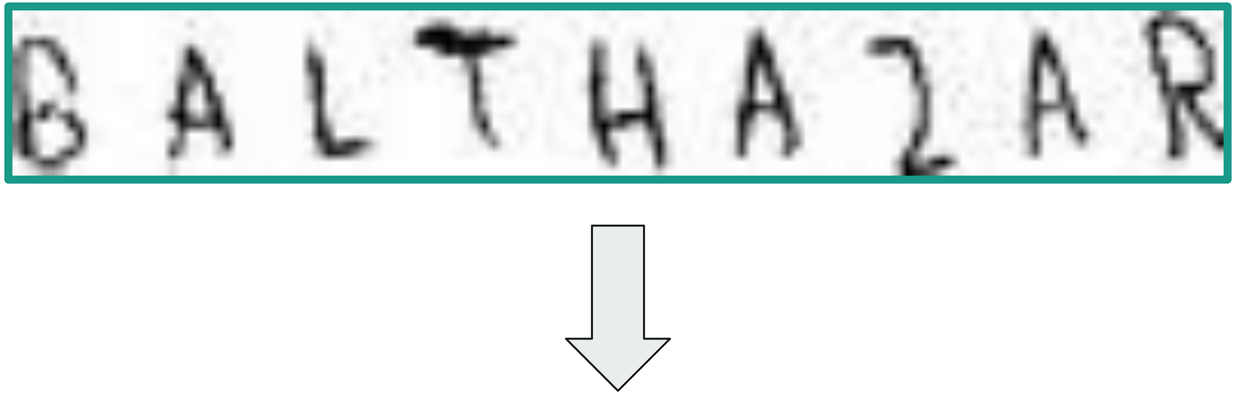
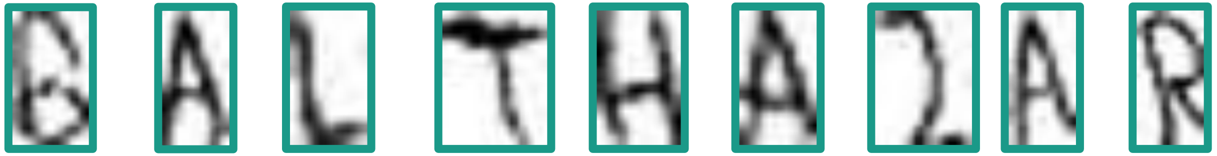

PRENOM : MARTIN





Done with Google Vision API[1]

Preprocessing (Letter Segmentation)

CRNN approach (Initial model)

CRNN approach used because it removes the need for character segmentation to train the model

Preprocessing:

- Removing images with null labels
- Removing images with the ‘Unreadable’ label
- Converting all labels to uppercase for uniformity as all images are in handwritten in uppercase
- Cropping all images to 256 x 64 for uniformity
- Normalising image values to be between 0 and 1
- Reshaping input array to be fed into the CNN

CRNN approach (Initial model) continued...

Notes about dataset used at this stage:

- The dataset still had significant noise (words in the image that were not in the labels)
- All of the dataset could not be used as loading it all into the memory at once required 40.4 GiB to store
- 30,000 training and 3,000 validation images used

CRNN approach (Initial model) continued...

Connectionist Temporal Classification (CTC) loss [3]

- The backbone of this approach
- Removes the need to annotate the dataset for each timestep
- Helps calculate the loss to train the model
  - Sums up all scores of all possible alignments (paths) of the Ground Truth label, this means that it does not matter where the text appears in the image
  - Take the log of this sum and put a minus in front of it, the function minimises this (loss) value
- Decodes the output matrix of the NN to get the text in the input image
  - Calculates the best path by taking the most likely character at each time-step
  - Undoes the encoding by removing duplicate characters and ctc blanks from the path revealing the recognised text
  - Approach is called best path decoding

CRNN approach (Initial model) continued...

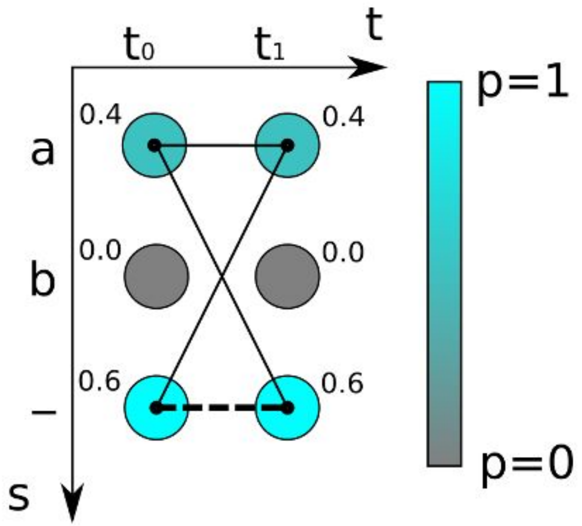


Fig. 3: Output matrix of NN. The character-probability is color-coded and is also printed next to each matrix entry. Thin lines are paths representing the text “a”, while the thick dashed line is the only path representing the text “”.

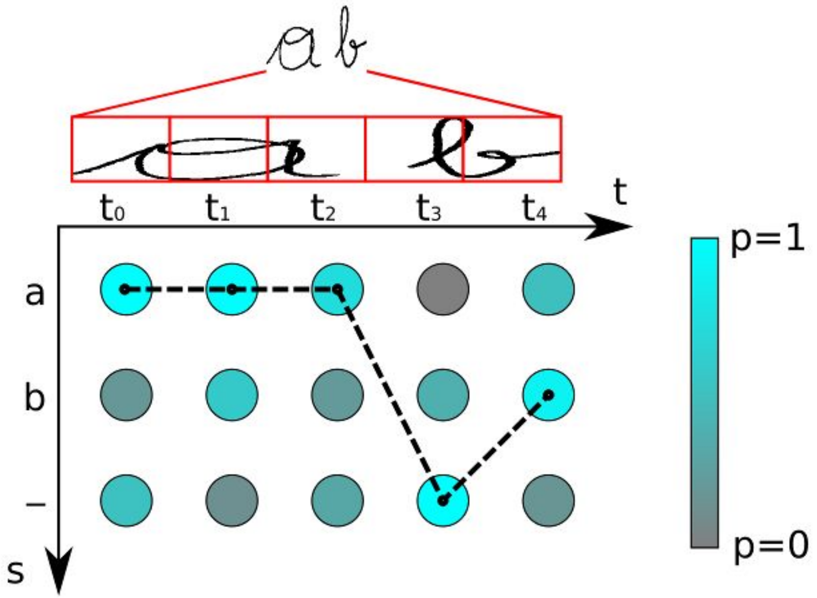


Fig. 4: Output matrix of NN. The thick dashed line represents the best path.

CRNN approach (Initial model) continued...

Model:

- 3 Conv2D layers from tf.keras.layers
    - A ReLU activation function is applied along with BatchNormalisation and a MaxPooling2D function. The 2nd and 3rd convolving layers alo have a dropout of 0.3 to lessen overfitting.
- The output is reshaped from here to be fed into the RNN
- 2 Bidirectional LSTM layers are used. We use bidirectional because literature suggests they help train the model better. In the final report, we might replace the LSTM layers with GRU layers for faster training times.
- The output of the model is a (64, 30) matrix. We have 64 timesteps for each image and the NN outputs the probability for each character at a timestep.

Results:

- 87.77% character accuracy and a 73.63% word accuracy

Improvements made to the CRNN approach

Preprocessing:

- Using the cleaned dataset for training
- Dropping images from the csv files that were corrupted

Training:

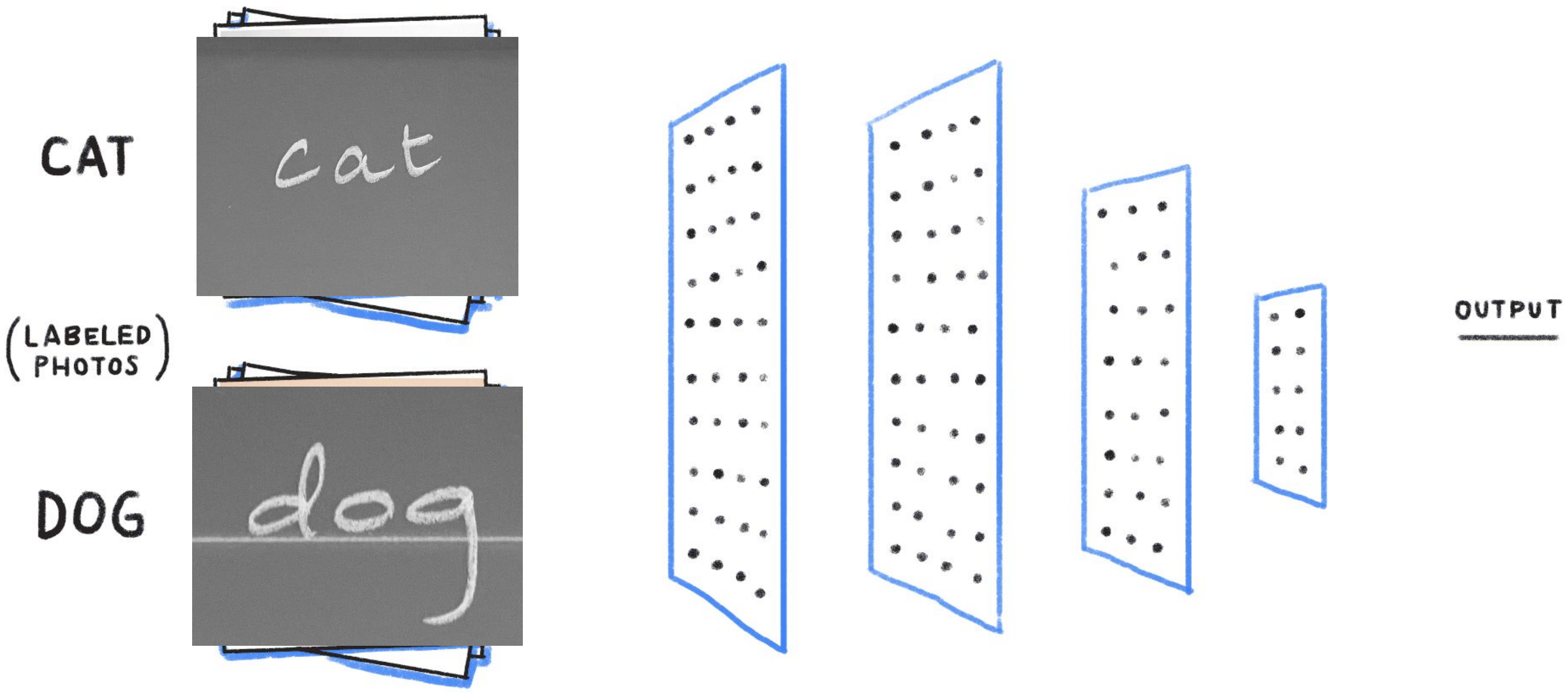
- Using a mini-batch training approach to avoid having to load the whole dataset in the memory, done by using python generators and a different approach using for loops. Comparisons will be made in the final report.
- Configuring tensorflow-gpu for faster training times by making use of the GPU
- Using early stopping to halt training if val_loss does not improve in 10 epochs, this can help avoid overfitting.
- Using model checkpoints to save the model weights that lead to highest accuracy and lowest loss on the validation set

Results will be shown in the final report with more detail, they are not available right now because the model is still training.

SVM Model

Preprocessing:

- Cropping images to content for all datasets
- Letter segmentation of train, test and validation datasets
- Resizing image to 32 x 32 since SVM takes input of same size
- Converting image to pixels
- Flattening each dataset
- Passing the data into a numpy array to be used for the SVM model


CAT
cat
(LABELED PHOTOS)
DOG
dog
OUTPUT

SVM

Gamma : defines how far the influence of single training examples reaches values leads to biased results.

Used as 0.001

The model used 200,000 images of letters for training data and 60,000 images of letters for testing data that was fit into an SVM model

Highest Accuracy observed was 61%

Assumed to be more if entire dataset is processed

kNN Model

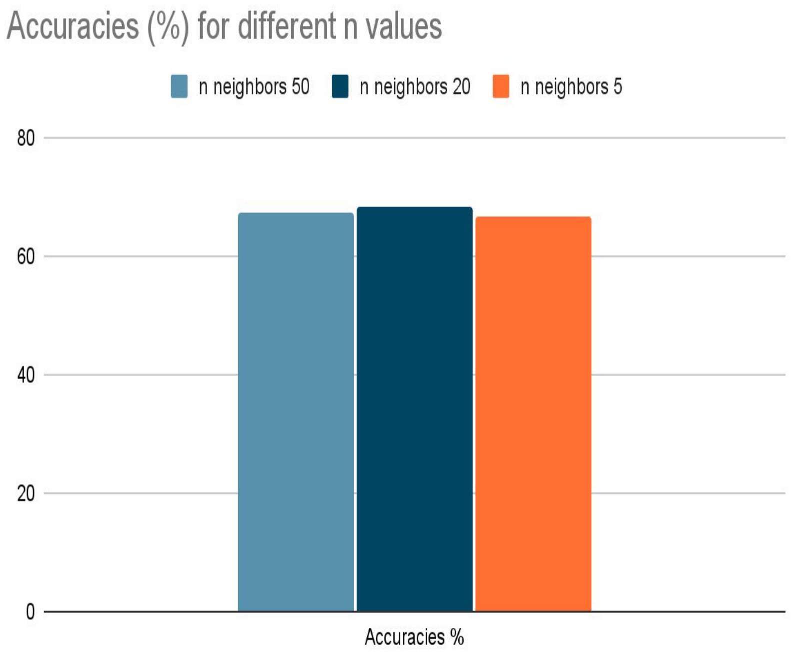
200.000 cropped images of letters used for training data.

40.000 cropped images of letters used for test data.

The n that gives best accuracy is 20 for our test data.

On average, %68,2 of the letters are predicted truly.

Assumed to get more accuracy results if entire dataset is processed



Gaussian Naive Bayes Model

Used 200.000 images with segmented letters.

Model is still in training.

Based on our research, we expect worse performance than all the other models we tried.

References

[1]. https://towardsdatascience.com/the-best-machine-learning-algorithm-for-handwritten-digits-recognition-2c6089ad8f09

[2]. https://www.kaggle.com/landlord/handwriting-recognition

[3]. https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c

[4]. https://medium.com/analytics-vidhya/image-classification-using-machine-learning-support-vector-machine-svm-dc7a0ec92e01

[5]. https://iq.opengenus.org/gaussian-naive-bayes/