EVOLVING HANDWRITTEN WORDS, A WORD AT A TIME

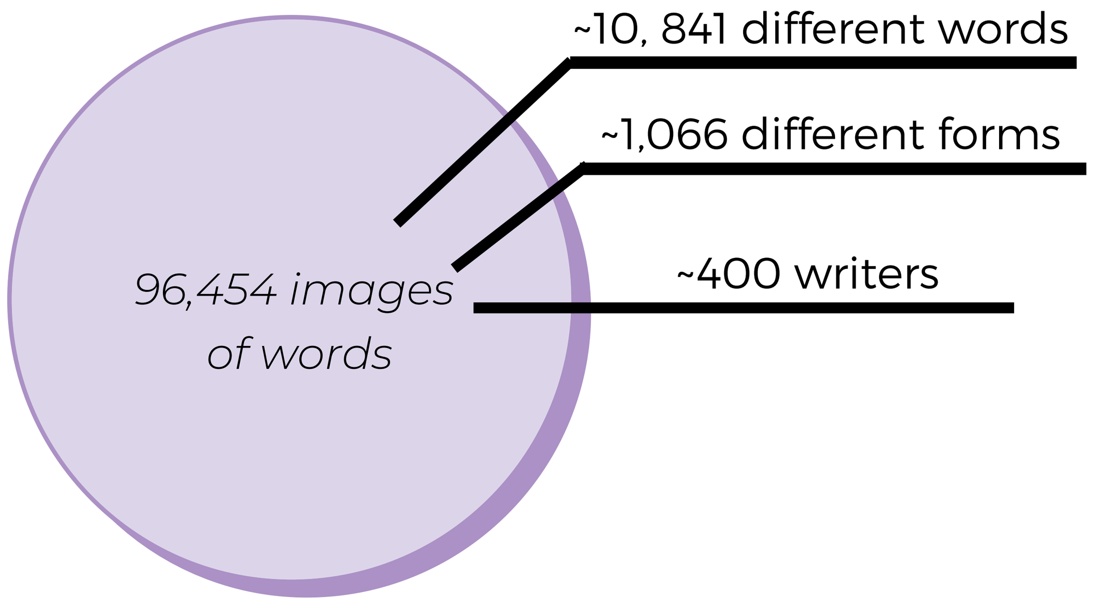
3244-2010-0006

Dannica Ong (e0202849@u.nus.edu)
Ignatius Ong (e0324134@u.nus.edu)
Chan Wen Yong (e0415677@u.nus.edu)
Low Hon Zheng (e0424726@u.nus.edu)
Chew BangYao Paul (e0424739@u.nus.edu)
Ge Wai Lok (e0323018@u.nus.edu)

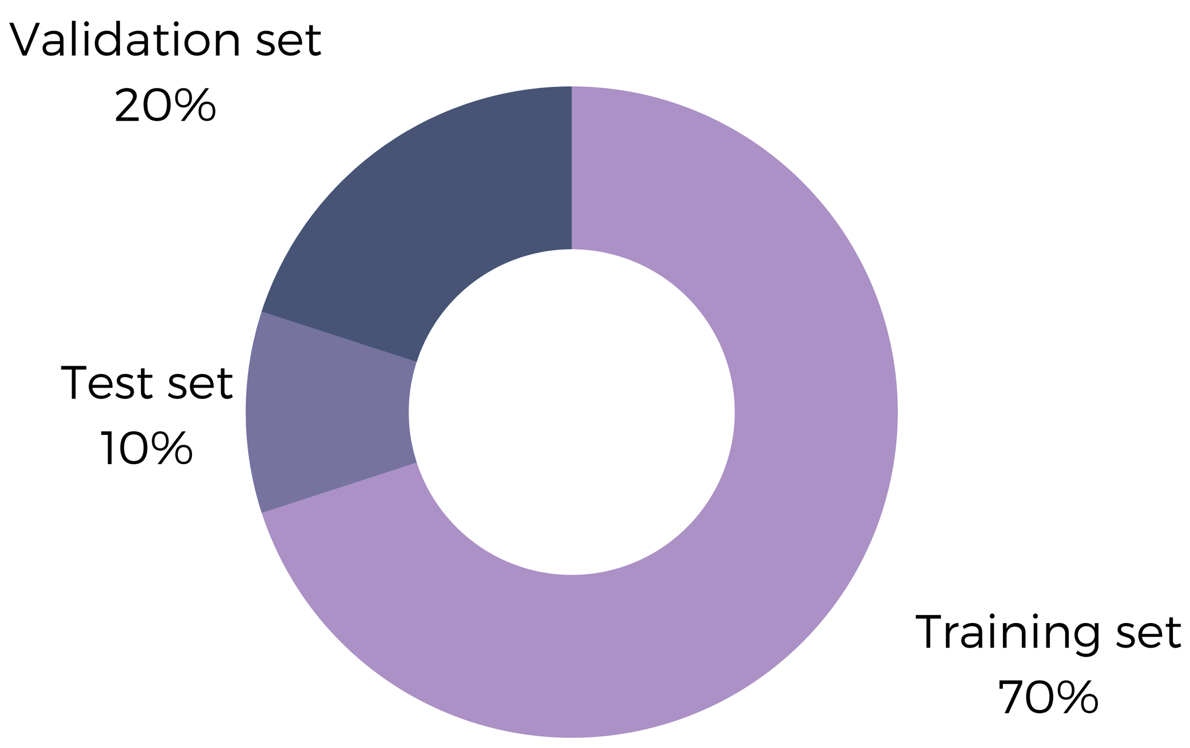
ABSTRACT

In the current working environment that we are in, we cannot escape the fact that we have to deal with tons of paperwork everyday. As such the different kinds of handwriting on these paperwork might be a hassle to read and understand.
Hence, this project aims to provide a solution to the above stated problem, by coming up with a model that can transcribe handwritten texts into standardised digital texts.

DATA



IAM dataset breakdown

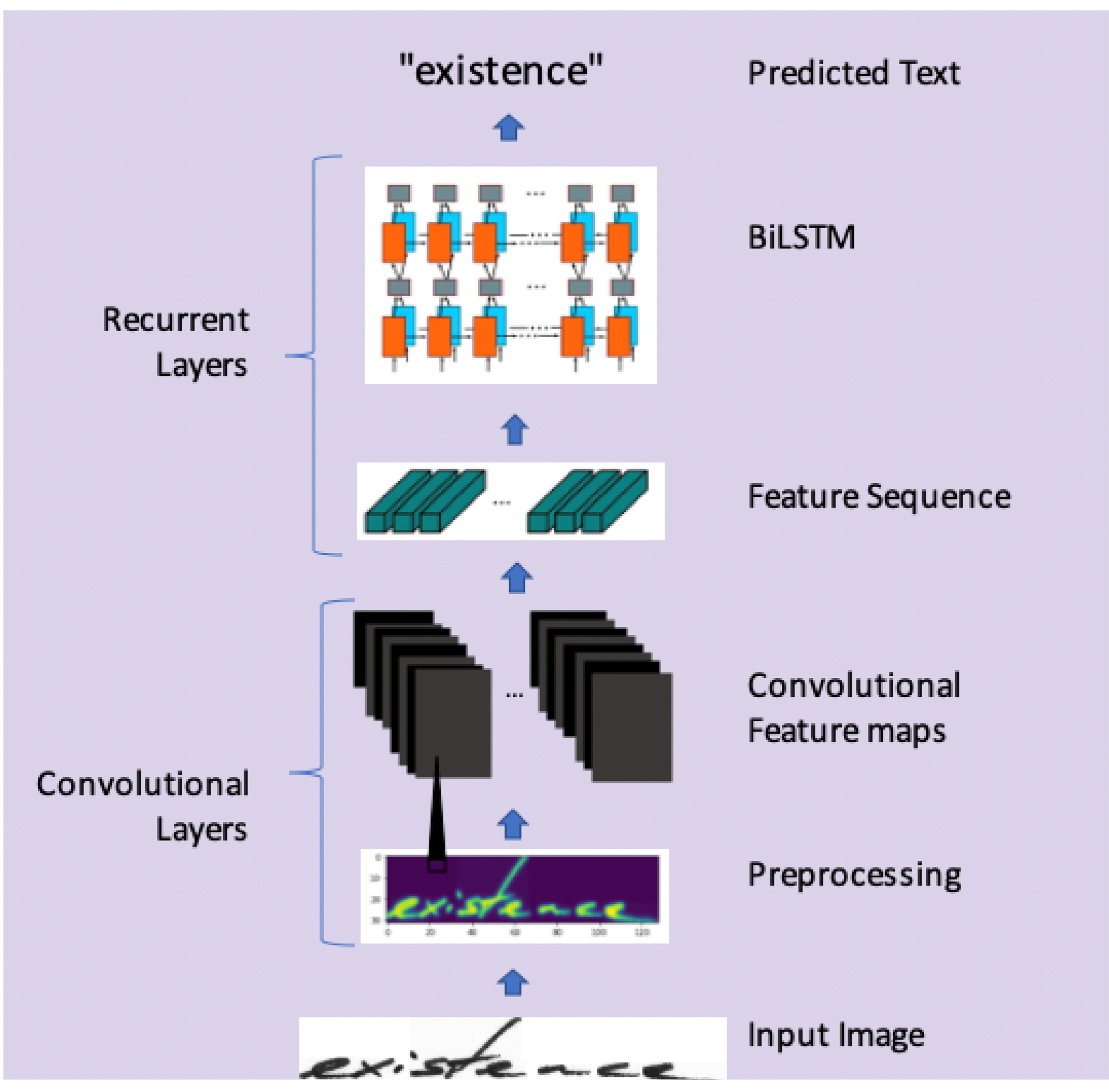


PROPOSED MODEL

*Given an input word image array, **x**, of size 128x32, with ground truth **y**, we want to approximate a function:*

h: x ⟶ y** such that **h(x) = y

Based on past project references, our models adopt a general structure shown below. Convolutional layers are used to extract features and Recurrent layers are used for classifying sequential data like text.



Proposed Model Structure

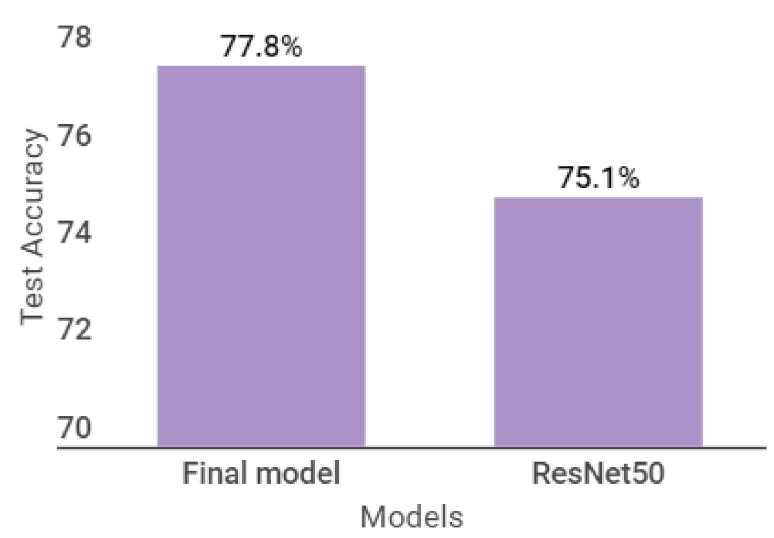
EVALUATION METRICS

$$Accuracy = \frac{Correct\ Predictions}{All\ Predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

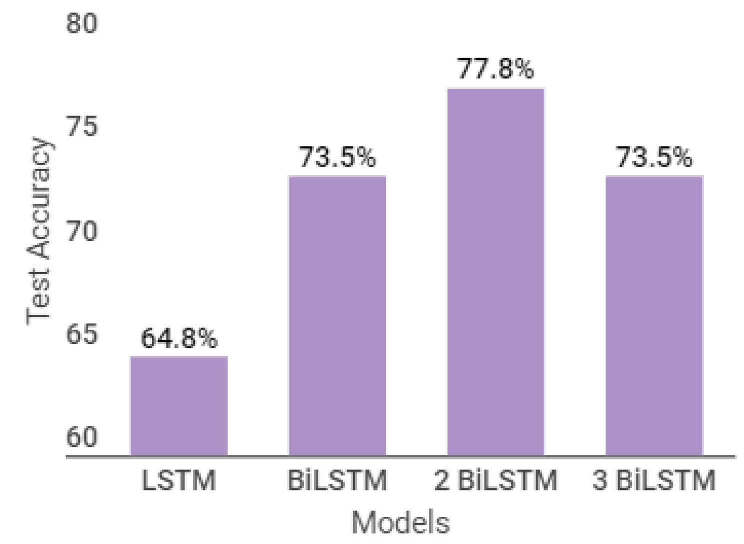
To evaluate the performance of the different models, we measured the accuracy of the model on a test set.

EXPERIMENTS AND DISCUSSION

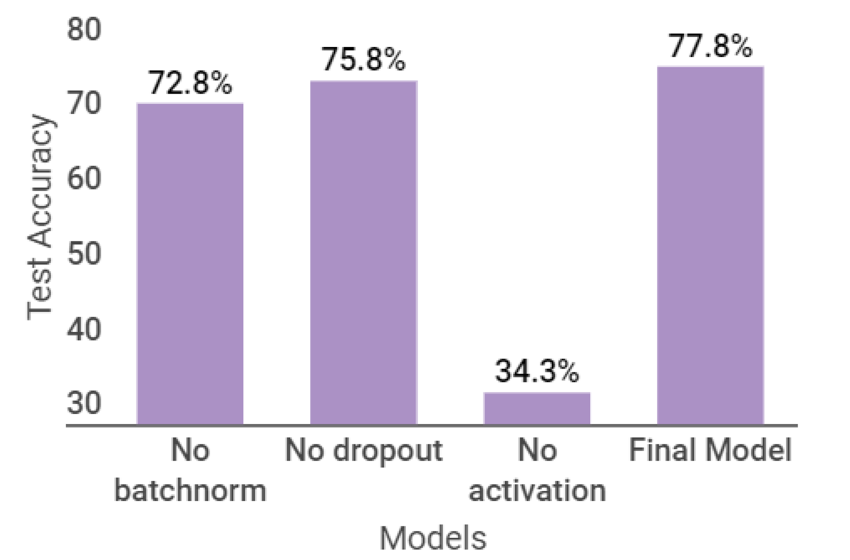
Keeping parts of the model and hyperparameters constant, changes were made to the model one at a time. The models were trained on the same train and validation set and evaluated on the same test set.



Effects of CNN layers on test accuracy



Effects of RNN layers on test accuracy

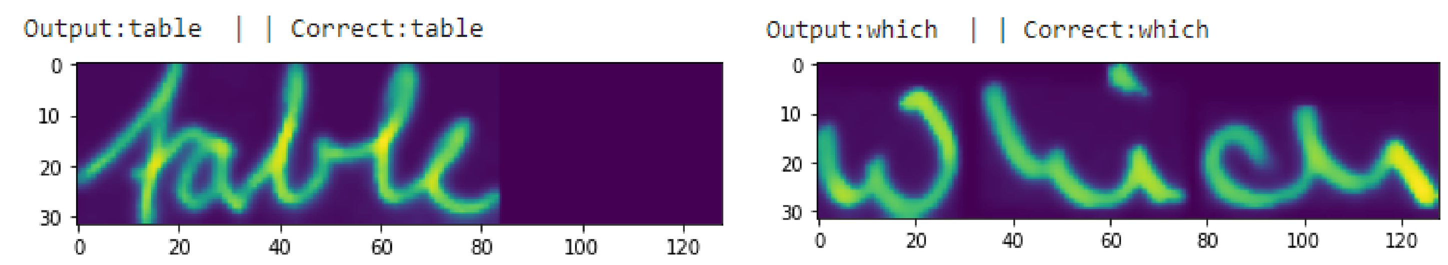


Effects of hidden layers on test accuracy

Left: *Using ResNet50 in place of 8 CNN layers seemed to have adverse effects on test accuracy despite the higher complexity. One reason could be the insufficient amount of data present, which resulted in the ResNet50 model overfitting more easily with its higher complexity.*
Middle: *We observed that using bidirectional LSTM in lieu of a normal LSTM significantly improved our model's accuracy. Our hypothesis is that Bi-LSTM allows information to flow in both directions compared to only one direction in LSTM, hence enabling our model to learn from information based on context from the past and the future.*
Right: *Dropout was used as a measure to tackle overfitting. Removing batchnorm had a negative effect on model performance. One reason could be that batchnorm allowed the model to converge faster and without it, more training epochs might be needed to achieve the same test accuracy. Removing activation also had a drastic effect on test accuracy as the model was unable to learn more complex functions without the non-linear transformations provided.*

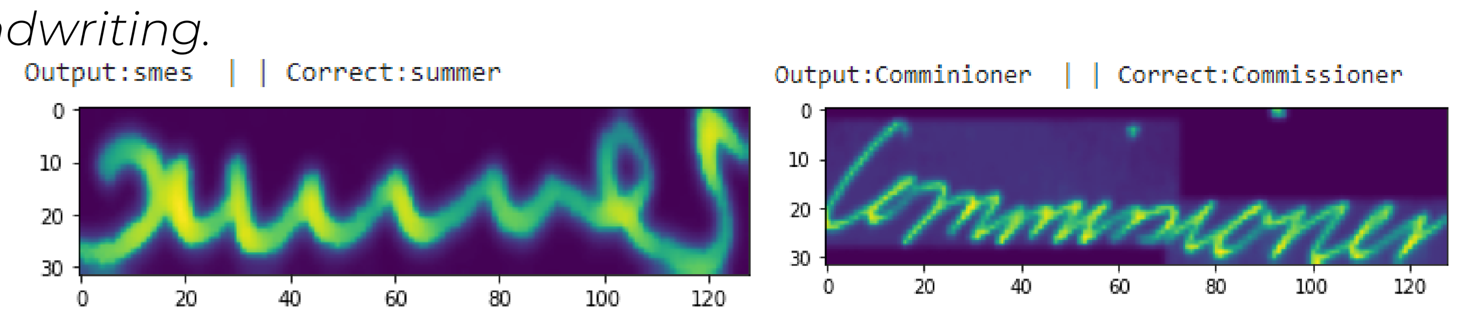
ERROR ANALYSIS

Surprisingly, our model was able to make out many words which even we as humans struggled to distinguish. This is a pleasant surprise as it aligns with our original motivation of recognizing bad handwriting.



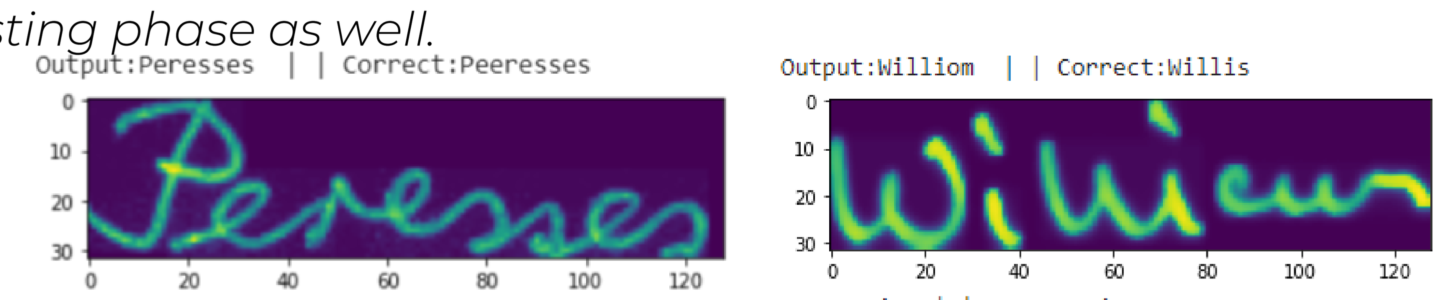
Some images that were correctly predicted

However, we observed that there were instances where the model was unable to correctly predict highly cursive handwriting.



Highly cursive or connected writing

In addition, upon further inspection, we also found poorly written or wrongly labelled images within the dataset. This could have affected our model during the training phase and subsequently it's accuracy in the testing phase as well.



Examples of noisy data

CONCLUSION

Our final model, with the highest test accuracy of 77.8%, is made of 8 convolution layers with batchnorm and dropout followed by 2 Bidirectional LSTM layers and an output layer. By attaching a spellchecker to our final model, it pushed the accuracy up to 81.0%. Here, we have shown that the model can perform decently on handwritten words that are less cursive in nature. Additionally, with just the IAM dataset, it appears that using a less complex network would be a better choice in feature extraction over more complex ones like ResNet50.

FUTURE WORKS

1. *Removing noisy data*
2. *Separating data into cursive and non cursive and train models specialised in each task*
3. *Using NLP to correct wrongly predicted words*
4. *Incorporating paragraph segmentation models*
5. *Exploring other accuracy metrics*

REFERENCES

1. *Madeyoga(2020) Handwritten-Text-Recognition - [online] Available at: https://github.com/madeyoga/Handwritten-Text-Recognition/blob/master/Preprocess%20%26%20Build.ipynb (Accessed: 4 Oct 2020)*
2. *Sundaram, G., Venktesh, B., Panigrahi, S. and Kapoor, V., (2020) Handwriting CNN LSTM [online] GitHub Available at: https://github.com/GireeshS22/Handwriting-CNN-LSTM (Accessed: 4 Oct 2020)*
3. *Sundaram, G., (2020) Handwriting To Text Conversion Using Time Distributed CNN And LSTM With CTC Loss Function. [online] Medium. Available at: https://towardsdatascience.com/handwriting-to-text-conversion-using-time-distributed-cnn-and-lstm-with-ctc-loss-function-a784dccc8ec3 (Accessed: 4 Oct 2020)*
4. *Dwivedi, P., (2019) Resnet_Keras [online] GitHub Available at: https://github.com/priya-dwivedi/Deep-Learning/tree/master/resnet_keras (Accessed: 4 Oct 2020)*
5. *Dwivedi, P., (2019) Understanding And Coding A Resnet In Keras. [online] Medium. Available at: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33 (Accessed: 4 Oct 2020)*