

<<<<

CTC LOSS FOR READING TEXT FROM IMAGES

Iago Casallerrey

ARTIFICIAL INTELLIGENCE (AI)

Dataset Creation and Image Preprocessing



Generating Letters

Using Arial font .ttf files, synthetic images of letters sized between 50 and 70 pixels were created. Image width was approximated based on letter count and font size.

OpenCV was used to crop images tightly, removing excess white space to speed up training and improve accuracy.

Augmentation Techniques

TensorFlow's ImageDataGenerator applied random rotations, zooms, shears, and brightness changes to increase dataset robustness.

This preprocessing helped the model generalize better to varied real-world inputs.

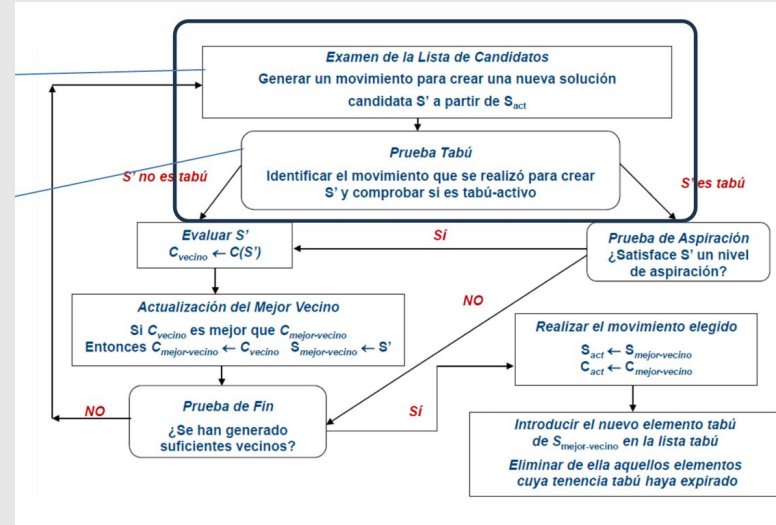
Creating a single letter recognition model with tabu search for latter transfer learning

Architecture

Multiple convolutional layers with ReLU activation extract key features, followed by max pooling to reduce image size and prevent overfitting.

Tabu Search metaheuristic optimized hyperparameters, notably learning rate, improving training speed and accuracy. I learned that reduced learning rate, leads to increased accuracy.

The model achieved 99% accuracy on test data, effectively recognizing individual letters with minimal errors.



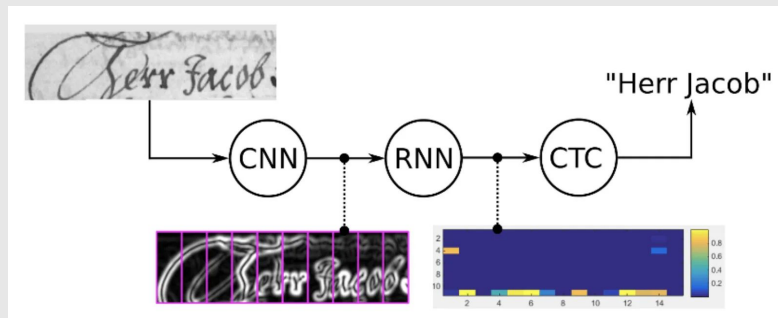
Creating a CRNN model for getting complete words

The model

The model uses convolutional layers together with recurrent LSTM that helps the model to remember past letters.

With that, it is able to read words like CIRUELA.

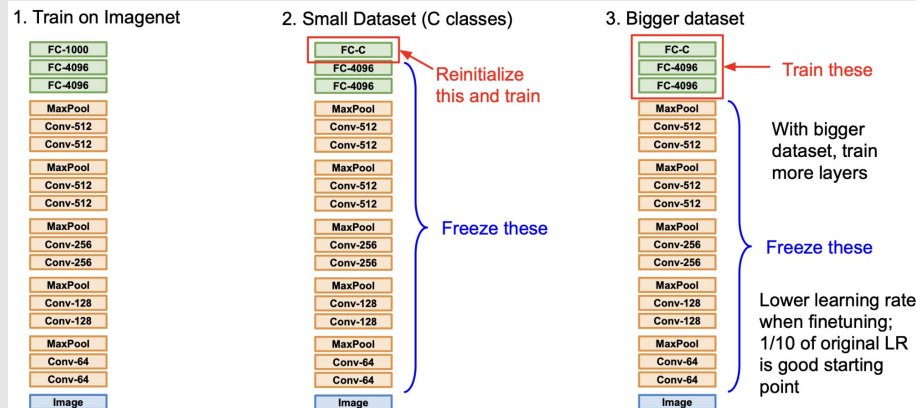
The loss function, connectionist temporal classification, divides the image in parts, and predicts for each part which letter. For example, if it read c c - i i - i r u e l a The result would be ciiruella, with 2 is, where the - are spaces in which the neural network learns to recognize empty spaces.



CIRUELA

Transfer learning the CRNN model with the Simple neural network model

The simple neural network and the CRNN model had a shared label with the simple neural network model. I simply, transferred the weights and biases from one layer to the other, hopping, that one neural network would learn from the other.

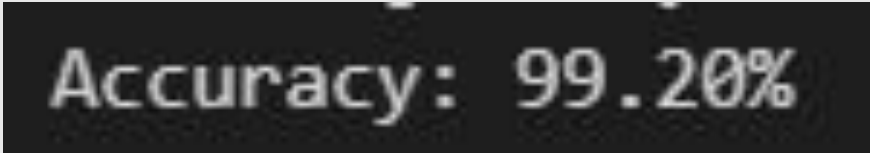


Model evaluation

The CRNN achieved 99.2% accuracy and a test loss of 0.0742 on the full dataset, demonstrating strong recognition capabilities.

The trained model was saved in a .pth file format for future fine tuning and deployment.

This enables reuse and extension of the model in subsequent projects.



Accuracy: 99.20%

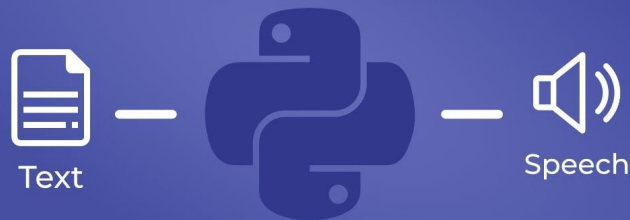
Text to speech for easiness to use

Output text from the CRNN is converted to speech using the gtts Python library, enhancing accessibility.

AudioSegment from pydub plays the generated speech, allowing repeated playback for user convenience.

This makes the result easier to hear, without making your eyes tired.

Convert Text to Speech



Comparison with hugging-face model

The hugging face model only has an accuracy of 96.4% in my dataset. My model has an accuracy of 99.2%. However, it has overfitting in 80% of the images.

My model is also quicker, but it may struggle with non arial images.

This makes an improvement in velocity and accuracy in Arial letters.