Deep Learning basics - Project

Phase 3 by Carmen Azorín

Handwriting recognition model

- The objective of the model is to recognize handwritten words.

Datasets:

- Training dataset: IAM Words
- Test dataset: My own dataset (132 words)

Loss function:

- CTC Loss function from Keras

Performance metric:

- CER (character error rate)
- Correct characters predicted
- Correct words predicted

Model overview

Convolutional neural network

- Convolutional layer
- Max pooling layer
- Dropout layer

Recurrent neural network

- Bidirectional LSTM

Dense layer + Softmax

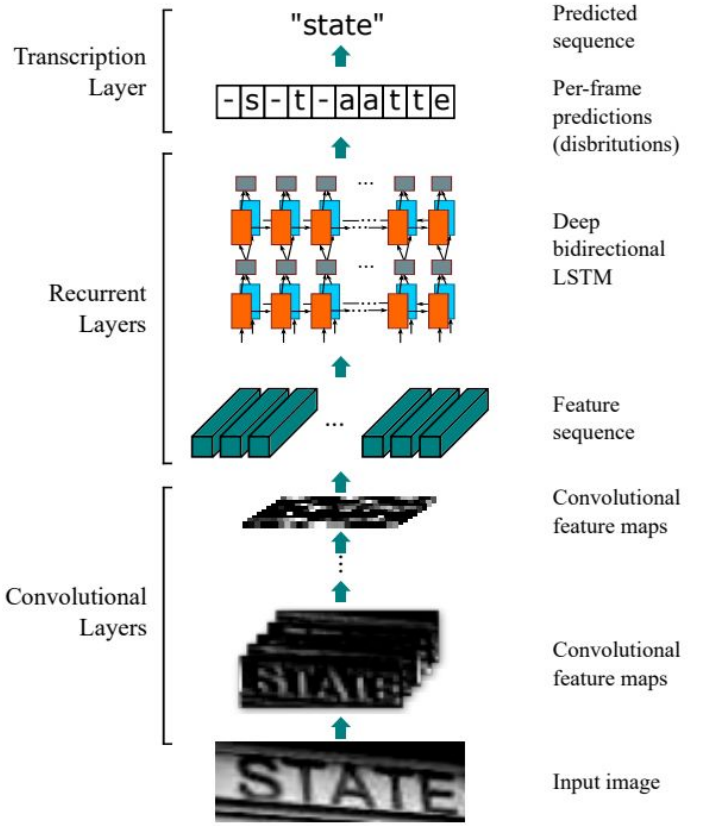


Figure 1. The network architecture. The architecture consists of three parts: 1) convolutional layers, which extract a feature sequence from the input image; 2) recurrent layers, which predict a label distribution for each frame; 3) transcription layer, which translates the per-frame predictions into the final label sequence.

Challenges faced

- For small length of the training data, the time was relatively short.
- As the size of the training data increased, the execution time increased considerably.
- With other values for the learning rate hyperparameter, the time didn’t improve.
- With Adadelta optimizer, the time got worse.

Final learning rate: 0.0001 Adam

Epoch	Time	Loss	Val loss
1	46s	11,61	32,77
2	36s	11,09	32,72
3	36s	10,63	37,76
4	35s	10,28	88,50
5	35s	10,02	34,72
6	32s	9,65	39,32
7	33s	9,38	67,62
8	32s	8,96	38,72
9	33s	8,57	37,18
10	32s	8,20	38,55
11	32s	7,63	40,43
12	33s	7,28	42,00
13	32s	6,88	50,97
14	33s	6,37	42,77
15	32s	5,99	46,48
16	33s	5,50	43,40
17	32s	5,04	45,94
18	33s	4,75	45,23
19	32s	4,53	88,25
20	33s	4,25	47,21

Challenges faced

- Loss decreases considerably without data augmentation.
- The problem comes when trying to reduce validation loss.
- We tried data augmentation and changing dropout hyperparameter to 0.5.

Epoch	Time	Loss	Val loss
3	137s	13,90	24,75
6	175s	12,83	24,86
9	136s	11,83	26,67
12	135s	10,63	27,21
15	136s	9,70	30,42
18	136s	8,55	30,72
21	139s	7,32	33,45
24	139s	5,94	38,30
27	141s	4,55	36,86
30	135s	3,28	40,68
33	135s	2,27	43,48
36	136s	1,53	47,64
39	136s	0,99	53,91
42	135s	0,75	54,27
45	135s	0,64	55,01
48	136s	0,46	53,69
51	135s	0,38	55,49
54	136s	0,28	49,71
57	136s	0,47	57,20
60	136s	0,75	64,17

Challenges faced

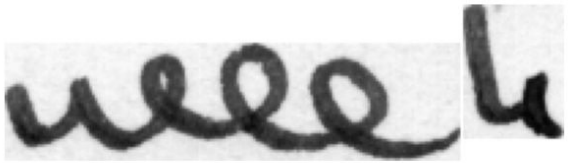
- Finally, I tried using other validation data, not my own.
- Results got better but not enough.

Performance metrics:

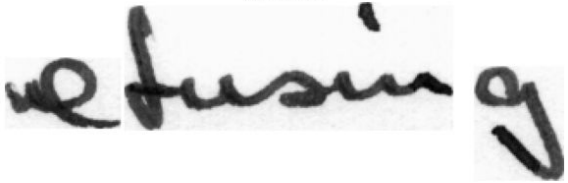
```
Correct characters predicted : 23.50%

Correct words predicted      : 19.18%

Character Error Rate         : 80.82%
```

Ntorh

iwech



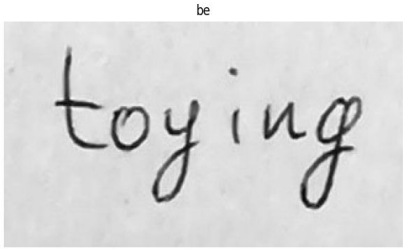
iprealuice

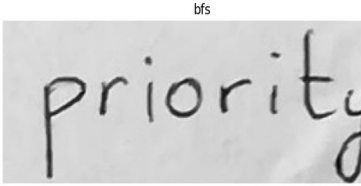


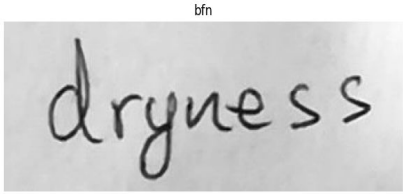
with

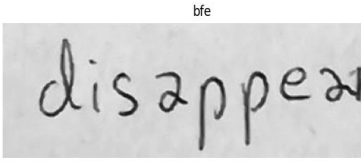
Best execution

- After testing the model with many different hyperparameters, I got the best run of my model with my own test data:
- Epochs = 60
- Batch size = 128
- Learning rate = 0.001 Adam










```
Correct characters predicted : 2.79%
Correct words predicted      : 0.00%
Mean Character Error Rate    : 100.00%
```

Conclusions

Since phase 2, I have collected more images for the test dataset. Specifically, I now have 132 images.

I have also tried changing values of new hyperparameters, such as dropout to 0.5. In addition to the number of epochs and the batch_size.

The best execution is still the one I delivered in phase 2.

With this project I have learned to differentiate the types of neural networks, I have improved my Python skills considerably and I have learned new libraries. But above all, I have put into practice everything I have learned in the theory of the course.

Although the results have not been as expected, I am proud of the work I have done.