IN5400 Mandatory 2

Espen Lønes (espenlon)

April 19, 2022

The source code is located in the 10 python files.
The 'cocoSource_xcnnfused.py' file is only used by the utils.model.py script to get loss_fn function.
'Exercise_Train_an_image_captioning_network_test_taskX.py' (X for task number). Holds the parameters as well as runs all necessary scripts to train and save a model.
'cocoSource_xcnnfused_taskX.py' files holds the code for the Image caption, RNN and Cell type classes.
The 'Exercise_Train_an_image_captioning_network_test_load.py' file contains the code for loading the pre trained model (model 4 / Task 4) and calculates validation error and makes predictions on a few images.

The code assumes it is located at an uio ml node. This can be done with the flowing command:
$ scp path/to/file/on/your/computer your_username@ml6.hpc.uio.no:.
The programs are then run using:
$ CUDA_VISIBLE_DEVICES=X python yourscript.py
where X is changed to a given gpu number.

Scripts also assumes they is in the same folder as the pycocoevalcap, utils and storedModels_test folders.

The trained model is located in the storedModels_test folder.
The root for the dataset is set to be '/itf-fi-ml/shared/IN5400/dataforall/mandatory2/data/coco/'.
Model 1, 2, 3 and 4 where trained for respectively 100,80,80 and 50 epochs, with a batch size of 128.
All other parameters except number of RNN layers, 'featurepathstub' and 'cellType' where the same for all models. These parameters can be found in the 'Exercise_Train...' files.
'Exercise_Train..' outputs; BLEU-4, Cider, ROUGE and METEOR score per epoch as well as loss.

For Task 1 the best epoch was 75 with METEOR score 0.24 and BLEU-4 score 0.27:
All validation output for best epoch:

```
{'testlen': 41787, 'reflen': 45714,
'guess': [41787, 36787, 32138, 27497],
'correct': [29477, 14847, 6988, 3388]}
ratio: 0.9140963381020932
Evaluation results, BLEU-4: 0.2690178662543596,
Cider: 0.7750904920588869,
ROUGE: 0.5012589134149434,
Meteor: 0.23804474171695475
```

For Task 2 the best epoch was 61 with METEOR score 0.24 BLEU-4 score 0.27:
All validation output for best epoch:

```
{'testlen': 42496, 'reflen': 46294,
'guess': [42496, 37496, 32884, 28277],
'correct': [29894, 15047, 7136, 3548]}
ratio: 0.9179591307728666
Evaluation results, BLEU-4: 0.27077970319262706,
Cider: 0.786723679007084,
ROUGE: 0.5019696308189218,
Meteor: 0.24044973215719312
```

For Task 3 the best epoch was 74 with METEOR score 0.24 BLEU-4 score 0.27:
All validation output for best epoch:

```
{'testlen': 39958, 'reflen': 45437,
'guess': [39958, 34958, 30442, 25936],
'correct': [28833, 14681, 7007, 3502]}
ratio: 0.8794154543653657
Evaluation results,
BLEU-4: 0.2716075220507252,
Cider: 0.7889710297103575,
ROUGE: 0.49874034113812177,
Meteor: 0.2377817058127434
```

For Task 4 the best epoch was 45 with METEOR score 0.24 BLEU-4 score 0.27:
All validation output for best epoch:

```
{'testlen': 42404, 'reflen': 45927,
'guess': [42404, 37404, 32749, 28100],
'correct': [30226, 15375, 7152, 3402]}
ratio: 0.9232913101225658
Evaluation results,
BLEU-4: 0.273023565785813,
Cider: 0.7915822516230098,
ROUGE: 0.5058655862529884,
Meteor: 0.24124189159923892
```

Comparing these results we see that we managed to train all models to similar performance. With simple rnn and LSTM taking the most epochs, slightly 'faster' (in terms of epoch) the GRU model. And with the by far least epochs the attention LSTM. One important observation is that even though the more complex models generally used fewer epochs they use more time per epoch the more complex they get.

Now comparing the scores from the best epoch with those from the loaded model (Task 4) we get:

```
-- Output from 'Exercise\_Train...load.py' --

{'testlen': 40431, 'reflen': 45507,
'guess': [40431, 35431, 30915, 26407],
'correct': [29038, 14710, 6936, 3364]}
ratio: 0.8884567209440112
Evaluation results,
BLEU-4: 0.26798784128785996,
Cider: 0.7686067204932451,
ROUGE: 0.49700256261201575,
Meteor: 0.23581309414384657
```

We see here that the results are slightly different but relatively close as one would expect.

The plots below show the train/validation loss (left) and Meteor score (right) for the 4 tasks.

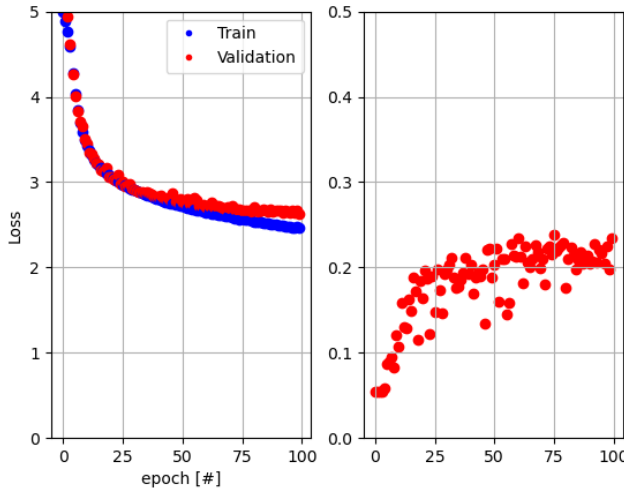


Figure 1: Simple one layer RNN

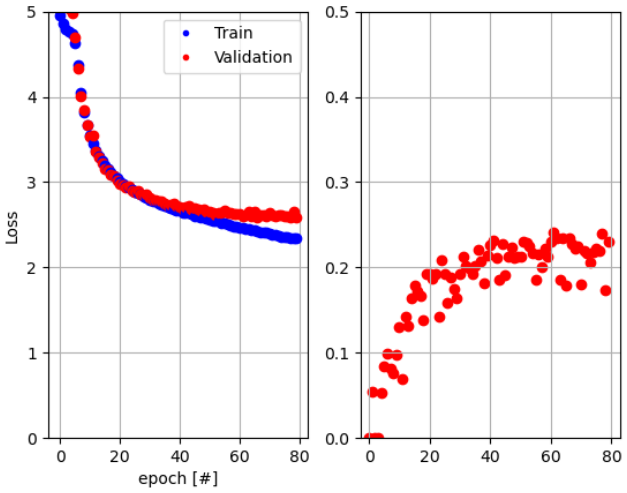


Figure 2: Two later GRU

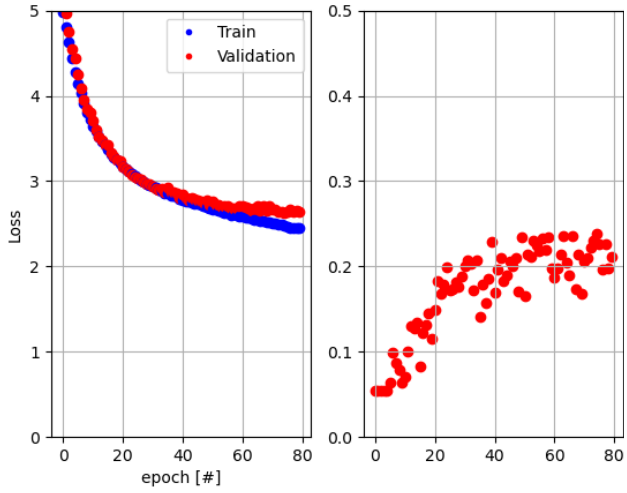


Figure 3: Two later LSTM

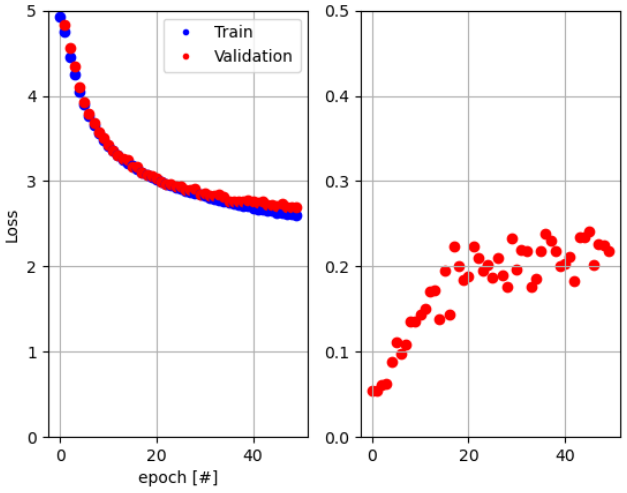
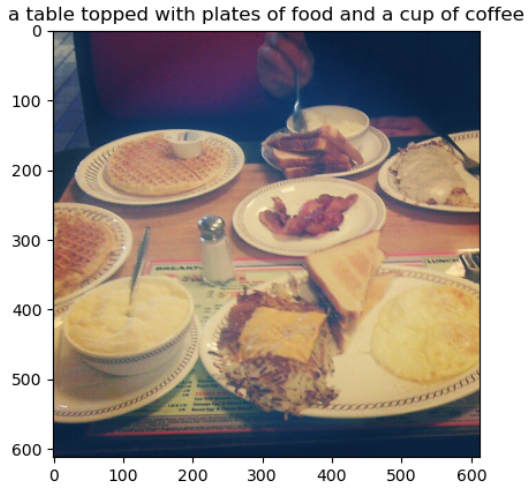
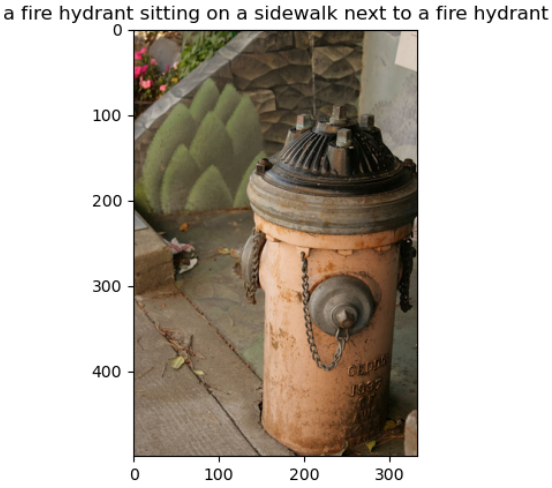
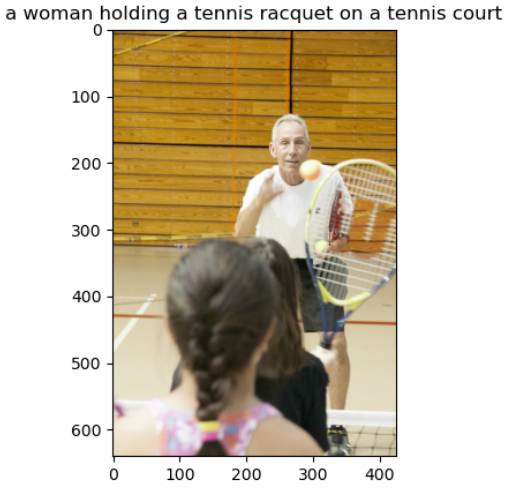


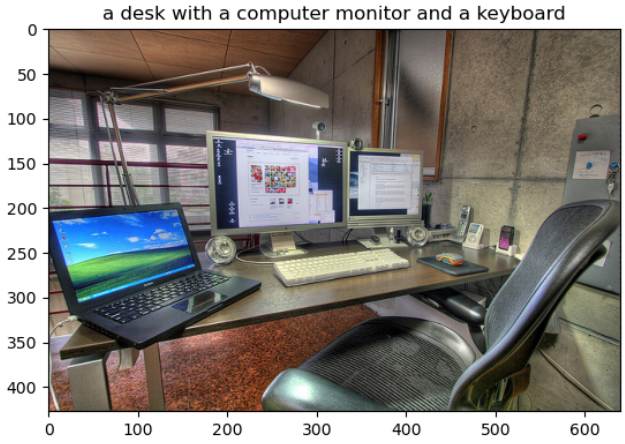
Figure 4: Two later Attention LSTM

Below are 5 Example images with suggested caption made by model 4 (attention LSTM):









We see that the model gets a lot of details right but it's far from perfect.