Deep Learning Lab

Exercise 2

Task 1

Head	IoU
Linear	0.42
Convolutional	0.48
Transformer	0.62
Transformer (shared Q&K)	0.59

1.2

The convolutional head performs slightly better. This could be because of the fitting choice of a convolutional layer for an image. It gives the model a slight advantage while using a similar amount of parameters

1.3.1

A transformer with shared queries and key performs slightly less worse than a normal transformer. However it also has much less parameters, so this decrease in performance is justified. Also it still performs much better than a linear or convolutional head.

Visually they don't look much different. When the normal transformer tends to color too much the Shared-Queries model tends to color not enough. If a smaller model is desired the shared queries-keys seem like a valid option.

1.4

Linear layer performs the worst, even worse than a convolutional layer when they have a similar amount of parameters. This could be because of the fitting choice of a convolutional layer for an image. It gives the model a slight advantage.

The transformer models perform much better but also have allot more parameters. They should be chosen if the outcome is important and there are many resources to spend.

Overall all heads scored quite well should be due to the large prefitted model that they extend.

By analyzing the plots, it can be shown, that the transformers learn less smoothly compared to the linear layer.

Plots and Images of the Linear Head

Plots and Images of the Convolutional Head

Plots and Images of the Transformer Head

Plots and Images of the Transformer Head with shared Q&K

Task 2

I had done the tasks before the error in the bleu-calculator was fixed and because of energy concerns I did not run all trainings again. All wrong scores are clearly marked to avoid confusion.

2.1

a. blib_captioning.py: "bleu": 0.406 (using incorrect way of calculating bleu)
   Notebook output
b. sampling:
   a. temp=1.0: "bleu": 0.156 (using incorrect way of calculating bleu)
   b. temp=0.7: "bleu": 0.238 (using incorrect way of calculating bleu)
   c. Why does the result improve with lower temperature?
      Having a higher temperature makes the output more exploratory and less accurate. In case of this training-setting this improves training but it is a hyperparameter and could be worse in a different scenario. In theory it makes sense to adjust the temperature over the duration of training (from exploratory to precise).
c. Experiments:
   (everything not mentioned was done as in step b) sampling)
   - Experiment 1
     - Prompt: “The image introduces the viewer to”
     - Bleu: 0.046 (correct score)
   - Experiment 2
     - Prompt: “The viewer comes upon a strange view of”
     - Bleu: 0.093 (correct score)
   - Experiment 3
     - Prompt: “an image of”
     - Bleu: 0.260 (correct score)
   - Results:
     - long prompts seem to perform worse than the short version
     - for some reason the influencing of experiment 2 seems to work slightly better

2.2

a. train_retrieval: "i_r1": 0.427
b. finetune: “i_r1”: 0.591
   - the score is better. The difference can easily be explained by using a prior fitted network. The network was already fitted on numerous images and is only adjusted to the current images. This gives a huge start-advantage.
c. Experiments:
   Goal: I wanted to see how the temperature affects the “normal” training from scratch
   - Experiment 1: temperature=0.5
     - “i_r1”: 0.324
   - Experiment 2: temperature=0.3
     - "i_r1": 0.385
   - Experiment 3: temperature=0.2
     - "i_r1": 0.429
   - Result:
     - by training from scratch and with a constant temperature, a value between 0.3 and 0.0 seems to work best (as 0.2 outperformed 0.1 slightly)