In this exercise you will experiment with semantic segmentation, image captioning and image-text retrieval.

Submitting

Create a report as PDF (max 2 pages). Upload your report and code as zip to ILIAS, the zip should usually be less than 10MB in size if you adhere to the following rules:

Please do not upload model checkpoints (`.pth` files), tensorboard files (`events.out.tfevents...`) or parts of the dataset. Do not upload the jupyter notebooks with images in them (`.ipynb`), as they will be large. If you want to upload images, add them to the report.

Setup

Pool computers

If you want to train on the computers in the PC pool, make sure you applied for an increase in disk space quota. Otherwise, the space in your home folder will not be enough.

How to use the pool:

- Connect to the login server of the technical faculty: `ssh username@login.informatik.uni-freiburg.de`
- Do not run anything on the main server. Connect to one of the pool computer there instead. Preferably use tfpool25-46 or tfpool51-63 as they provide the best GPUs. `ssh tfpoolXX` - replace XX with a 2 digit number of the computer you want to connect. Before running make sure no one else is using the selected computer and the GPU is free.
- See GPU load: `nvidia-smi`
- See logged in accounts: `who`
- View running processes: `top`

Start a screen session to keep your code running in the background:

- Start the session
```
screen -S t01
# run your experiment
```
- Detach from screen using: ctrl+a+d
- Login back into screen:
```
screen -ls
screen -r t01
```
- Write down on which computer you started your screen session.
- ctrl+esc to enter copy mode if you need to scroll up, navigate with arrow/page keys, copy output etc

Installing python using miniconda

We recommend installing python inside a conda environment, since this way the installation will be independent of the python installed on the system. Below are the installation instructions for Ubuntu, on other operating systems the workflow is similar.

- Download the latest Miniconda3 Linux 64-bit installer: `https://docs.conda.io/en/latest/miniconda.html` Usually the 64-bit version is correct. Use `wget [link]` to download a file in the terminal.
- Open the terminal and `cd` into the folder containing *Miniconda3-latest-Linux-x86_64.sh*.
- Run `bash Miniconda3-latest-Linux-x86_64.sh` to install it into your home directory. When asked by the installer to initialize Miniconda3, answer yes. This way, your `~/.bashrc` file will be modified so you have conda available at startup.

- Close and reopen your terminal window to make sure the changes take effect.
- Test your installation by running `conda --version` to see the its version.

Setup conda environment

Next we will set up conda with a new environment named `lab` which uses Python 3.9. Run the following commands in your terminal (macOS, Linux) or miniconda prompt (Windows):

```
conda --version
conda update -n base -c defaults conda -y
conda create --name lab python=3.9 -y
conda activate lab
conda env list
```

The last command should show you all installed environments (including `base` and the activated `lab` environment). This way, you have a separate environment to install everything for this course and avoid possible interference with/from other python projects.

Note: Whenever you start the miniconda prompt/terminal for this course, run `conda activate lab` to switch to the correct environment.

Setup the code

1. Make sure your conda environment is activated and you are using python 3.9.
2. Download the code from ILIAS and cd into the folder.
3. Install PyTorch with GPU support:
   `conda install pytorch torchvision pytorch-cuda=11.8 -c pytorch -c nvidia -y`
4. Install packages: `pip install -U -r requirements.txt`

Setup the data and model checkpoints

On the pool

If you are working on the pool you should already have access to the data and checkpoint. Run
`ls /project/dl2023s/lmbcvtst/public` to check if you can see the folder.

Locally

If you want to run the exercise on your own hardware you will need to setup the data and model checkpoint locally.

Uncomment the last two lines in `paths.py` to set the code to load from local paths **or** set the environment variables to the local paths.

```
# assuming you are in the root of the repository
# create data folder
mkdir data

# download and extract dataset into the data folder
wget http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval_11-May-2012.tar
tar -xf VOCtrainval_11-May-2012.tar -C data
```

```
# copy the captions into the data folder
cp -r datasets/Captions data/VOCdevkit/VOC2012

# download the DINO checkpoint for the segmentation task
mkdir ckpt
wget https://dl.fbaipublicfiles.com/dino/dino_deitsmall16_pretrain/dino_deitsmall16_pretrain.pth -P ckpt

# download the original BLIP checkpoint for the image-text tasks
wget https://storage.googleapis.com/sfr-vision-language-research/BLIP/models/model_base_capfilt_large.pth

# run the postprocessing script, this should create ckpt/blip_model_base.pth
python postprocess_ckpt.py

# optionally remove the downloaded files
rm VOCtrainval_11-May-2012.tar
rm model_base_capfilt_large.pth
```

1 Self-Supervised Learning

In this exercise you will experiment with the DINOv2 model.

- `ex_image_selfsup.ipynb`: Jupyter Notebook with exercises.

1.1 Nearest Neighbor Features with Negative Feature Space Distance as Score

Todo: Complete method `get_matches_feats_A_to_B_nn`. This method takes as input two feature maps. It returns the Nearest Neighbors for each feature of feature map A to the features of feature map B. Additionally, it returns the negative distance for each match in feature space as score. (1 Points)

1.2 Visualize Nearest Neighbors and Calculate Crop Error with Feature Distance Ranking

Todo: Add the cropped-matches-error plot into your report using the cats images with image ids [78, 288, 133, 415]. Also add the output for mean and median error for the top 10 matches. Describe the plot with your own words. (2 Points)

Note: To make a screenshot of the whole plot, right-click on the cell's output and click "Disable Scrolling for Outputs".

1.3 Nearest Neighbor Features with Negative Cycle Distance as Score

Todo: Complete method `get_matches_feats_A_to_B_nn_with_score_cycle_dist`. This method takes as input two feature maps. It returns the Nearest Neighbors for each feature of feature map A to the features of feature map B. Additionally, it returns the negative cycle distance for each pair in pixel space as score. (1 Points)

1.4 Visualize Nearest Neighbors and Calculate Crop Error with Cycle Distance Ranking

Todo: Add the cropped-matches-error plot into your report using the cats images with image ids [78, 288, 133, 415]. Also add the output for mean and median error for the top 10 matches. How does the cycle distance score performs against the feature distance score? (2 Points)

Note: To make a screenshot of the whole plot, right-click on the cell's output and click "Disable Scrolling for Outputs".

1.5 Complete Nearest Neighbors Matches for 1-to-Many Frames

Todo: Complete method `get_matches_frames_A_to_Bs`. This method takes as input one reference frame (A) and multiple source frames (Bs). It returns the Nearest Neighbors Matches for each feature of feature map A to the features of each feature map B (of Bs). Additionally, it returns the score for each matched feature of feature map A. This score is averaged over all scores for a single feature matched with multiple source feature maps (Bs). If `pca_k` > 0, map the features to a `pca_k`-dimensional subspace using PCA. (2 Points)

1.6 Visualize Nearest Neighbors Matches for 1-to-Many Frames

Todo: Add the nearest neighbor visualization for the reference cat image (id 78) with the source images (ids 288, 133, 415) to your report. Show four different methods.

- `get_matches_feats_A_to_B_nn` without PCA.
- `get_matches_feats_A_to_B_nn` with PCA (10 components)
- `get_matches_feats_A_to_B_nn_with_score_cycle_dist` without PCA
- `get_matches_feats_A_to_B_nn_with_score_cycle_dist` with PCA (10 components)

What is the advantage of using PCA? What is the advantage of using the negative cycle distance as score? (2 Points)

2 Image-Text

In this exercise you will experiment with the BLIP [1] model. The model is defined in the following folders:

- `ex_image_text.ipynb`: Jupyter Notebook with exercises, use this for evaluation.
- `models/bert/`: Text encoder and decoder.
- `models/vit/`: Vision encoder.
- `models/blip/`: The model which connects the vision and text parts. `BlipCaption` uses the vision encoder and text decoder for image captioning, `BlipRetrieval` uses the vision encoder and text encoder for image-text retrieval.

Run `jupyter notebook --port 8888` and open `ex_image_text.ipynb`

2.1 Image Captioning

Your first task will be to complete the inference code to generate captions for the given VOC dataset.

2.1.1 Complete Caption Generation with Greedy Search

Todo: In file `models/blip/blip_caption.py` complete the methods `generate` and `greedy_search`. Use the jypter notebook to generate and evaluate captions for the VOC dataset. You should get about 28% BLEU score. (2 points)

2.1.2 Complete Caption Generation with Sampling

Todo: In file `models/blip/blip_caption.py` complete the method `sampling`. There, Top-K sampling instead of greedy search is used to select the next token when decoding. Evaluate again with Top-K sampling, you should get a lower BLEU score of about 7% for temperature $\tau = 1.0$ and about 12% for $\tau = 0.7$. Why do the results improve with lower temperature? (1 point)

2.1.3 Prompt Engineering

Todo: Experiment with different prompts (the default prompt is "a picture of "). Can you improve the BLEU score? Try at least 3 new settings and note the prompt and the resulting BLEU score in a table for each setting. Add the table to your report. (1 point)

2.1.4 Student Hyperparameter Search

Todo: Experiment with different decoding parameters (Top-K with different K and temperature or greedy decoding). Can you improve the BLEU score? Try at least 3 new settings and note the hyperparameters, the prompt and the resulting BLEU score in a table for each setting. Add the table to your report. (1 point)

2.2 Image-Text Retrieval

Your second task will be to train and evaluate the retrieval head of the BLIP model.

2.2.1 Complete Forward Pass and Evaluate

Todo: Complete the forward pass in file `models/blip/blip_retrieval.py`. Evaluate your implementation with the provided checkpoint. You should get about 54% image-to-text R@1. (2 points)

2.2.2 Complete Loss and Train from Scratch

Todo: Complete the loss computation in file `train_retrieval.py` function `train_epoch`. Train the retrieval projection layers from scratch (i.e. from random initialization). You should get about 43% image-to-text R@1. (1 point)

Optional: start a tensorboard server `tensorboard --logdir outputs --port 6006` and watch the experiment in the browser.

2.2.3 Finetune instead of Train from Scratch

Todo: Now, try finetuning the head instead with `--finetune`. Set learning rate to 1e-5, weight decay to 0 and train for 3 epochs. What score do you get and how can you explain the difference to the score when training from scratch? (1 point)

2.2.4 Student Hyperparameter Search

Todo: Experiment with different hyperparameters in the random initialization setting (i.e. without finetuning). Try at least 3 new hyperparameter settings and note the hyperparameters and the resulting image-to-text R@1 score in a table for each setting. Can you improve over the baselines? Add the table to your report. (1 point)

Optional: Visualize Top 10 results for a search query.

Additional resources

If you have difficulties understanding PyTorch, we suggest to check out some of the tutorials for PyTorch, learn the basics is a good tutorial to get started.

A very important concept is indexing of tensors, see numpy indexing. The indexing in PyTorch works very similar to numpy.

We recommend using an IDE like VSCode or PyCharm for writing and debugging python code.

Using browser-based tools

In this exercise, you will **optionally** use browser-based tools like jupyter notebook and tensorboard. The easy way to do this is to run the server and browser on the same machine (either directly sit in the pool or setup everything on your home computer).

However, if your code is running on a *different computer as the browser* , i.e. you are running the code on a pool machine but running the browser on your home computer) you will need to: **1)** start the respective server on the pool machine. **2)** establish a SSH tunnel from your home computer to the tf login node. **3)** establish a SSH tunnel from the tf login node to the pool machine where the server is running. The tunneling command looks like this:

```
ssh -v -N -L ${port}:localhost:${port} "${user}@${targethost}"
```

You might need to change the port of your server application if the port is already in use by another person. Note that everyone who can access the login node will be potentially able to see your server. Jupyter notebook protects this with an access token by default, but tensorboard does not.

References

[1] Junnan Li et al. *BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation.* 2022. arXiv: 2201.12086 [cs.CV].