Project Idea 1: Image to image translation using cycle GANs

(TA: Rachneet)

Implement a Cycle-consistent Generative Adversarial Network (CycleGAN) model from scratch for image-to-image translation problems based on the paper [1].

This model can translate an input image to another image based on the model parameters. If the model parameters are trained on image collections of horses and zebras, then the model can translate an input horse image into an output zebra image in terms of color. If the model is trained on image collections of photos and art paintings, then the model can output an image in the art style of the input image. This technique is useful for paired images generation including collection style transfer, object transfiguration, season transfer, photo enhancement.

The datasets for the implementation are available at [2].

Paper: http://openaccess.thecvf.com/content_iccv_2017/html/Zhu_Unpaired_Image-To-Image_Translation_ICCV_2017_paper.html

Datasets:
https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/

Pytorch code:
https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

Another helpful link for implementation:
https://cyclegans.github.io/

Expectations:
Successful implementation on at least one of the datasets

- Visual results similar as in https://cyclegans.github.io/project1/2018/04/29/Results-and-Analysis-on-Cycle-GAN-Implementation/

- Loss function values similar to Tables 1-5 in paper [1]

Exploration Ideas:

- Hyperparameter and architecture exploration and tuning
- Implementing distributed training
- Speeding up the data loader (Try to vectorize instead of using for loops, load data on the fly rather than storing it in memory)
- Try to test on data that was never seen as part of training the model

Project Idea 2: Show and Tell - A Neural Image Caption Generator

(TA: Rachneet)

Build a generative model using latest techniques in Computer Vision and Machine Translation to describe an image. This project involves implementation of the paper [3]. In this project, a single joint model is constructed to generate a target sequence of words that describes the input image. The idea is similar to encoder-decoder RNN used in translating sentences except that a Convolutional Neural Network is used in place of the encoder RNN. CNN, which takes image as an input, is trained for an image classification task to generate a compact representation of the original image. This representation is then passed as an input to a decoder RNN that generates the sentences.

Paper:
https://arxiv.org/abs/1411.4555

Datasets:
MSCOCO 2014 [4] and Flickr30k [5]

Code:
https://github.com/tensorflow/models/tree/master/research/im2txt

Expectations:
Implementation on at least one of the datasets and performance evaluation using BLEU, CIDEr and ROUGE_L metrics similar to as follows:

Dataset	**BLEU1** **Our code | Paper**	**BLEU2** **Our code | Paper**	**BLEU3** **Our code | Paper**	**BLEU4** **Our code | Paper**	**CIDEr** **Our code | Paper**	**ROUGE_L** **Our code | Paper**
COCO	64.6 | 71.3	45.9 | 54.2	31.7 | 40.7	22.0 | 27.7	69.4 | 85.5	47.6 | 53.0
Flickr30k	53.9 | 63	34.8 | NA	22.0 | NA	14.3 | NA	26.5 | NA	39.5 | NA

Project Idea 3: Deep Image Prior

(TA: Yuanyi Zhong)

Deep Image Prior is a type of convolutional neural network used to enhance a given image with no prior training data other than the image itself. A neural-network is randomly initialized and used as prior to solve inverse problems such as noise reduction, super-resolution, and inpainting. Image statistics is captured by the structure of a convolutional image generator rather than by any previously learned capabilities.

Paper:
https://arxiv.org/abs/1711.10925
Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

Project page:
https://dmitryulyanov.github.io/deep_image_prior

Code:
https://github.com/DmitryUlyanov/deep-image-prior

Expectations:
Implement and compare at least 3 architecture choices. Choose 2 images, plot PSNR and visualize progress at different iterations. Run tests on the test images; compute PSNR on the standard data set used in the paper, should get approximately the same PSNRs (e.g. within 10% relative difference)

Project Idea 4: Distributional Deep Q-Learning

(TA: Yuanyi Zhong)

Distributional RL is an interesting recent development of value based RL, where the algorithm models the entire distribution of Q values, instead of only the expectation as in the usual Bellman equation. It seems to lead to great empirical improvements. This is an extension to the deep RL assignment.

Paper:
https://arxiv.org/pdf/1707.06887.pdf
Bellemare, Marc G., Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

Code:
https://github.com/google/dopamine

Helpful Blog post:
https://flyyufelix.github.io/2017/10/24/distributional-bellman.html

Expectations:
Implement the paper (the C51 algorithm) in pytorch (not tensorflow!) and test it with simple Atari games, e.g. Pong, Breakout, etc., and show reasonable scores (average >= 15 for Pong, >= 100 for Breakout).

Project Idea 5: BERT and XLNet

(TA: Peijun)

Study two papers:

1. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
2. Yang, Zhilin, et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding." arXiv preprint arXiv:1906.08237 (2019).

“XLNet” achieves state-of-the-art results on 18 tasks. To be precise, these 18 tasks are:

- Three reading comprehension tasks on RACE (Table 1) and SquAD 2.0 and SquAD 1.1 (Table 2).

- Seven text classification tasks on datasets as shown on Table 3.
- Seven GLUE language understanding tasks as shown on Table 4. “XLNet” does not achieves state-of-the-art results on QQP and CoLA, so we don’t consider testing these two tasks.
- One document ranking task on ClueWeb09-B Dataset as shown on Table 5.

The **goal** of this project is to compare “BERT” and “XLNet” by fine-tuning these two models **individually** on these 18 tasks.

You should compare the results of “BERT” and “XLNet” and see if they match with the ones in the tables of the “XLNet” paper. Note that “BERT” is not reported for some of the tasks in “XLNet” paper, but you still need to test “BERT” on these tasks and included the results in your report.

You should not only use the fine-tuning hyper-parameters provided in the papers, but also try different hyper-parameters (e.g. weight decay, learning rate decay, sequence lengths) and report at least two more experiments’ results based on your chosen hyper-parameters.

If you still have time and available GPU hours, you are highly recommended to explore more with “BERT” and “XLNet” by fine-tuning the models on some NLP tasks that you think are interesting or challenging.

In the report, you will discuss your fine-tuning process, the parameters you use, the comparison of “BERT” and “XLNet” and your understandings or findings on these experiments. For example, you might want to discuss why the models perform better under certain combinations of hyperparameters.

Description of BERT:

- BERT builds on top of a number of ideas that have been bubbling up in the NLP community recently such as Semi-supervised Sequence, ELMo, ULMFiT, the OpenAI transformer, and the Transformer.
- The model is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers.
- The paper mainly shows that BERT can be fine-tuning with just one additional output layer to create state-of-the-art models for a wide range of tasks.
- Besides fine-tuning, the pretrained BERT model can be used to create contextualized word embeddings for existing models.
- The paper shows that BERT for feature extraction yields results not far behind fine-tuning BERT on tasks such as named-entity recognition.

Description of XLNet:

- XLNet is a generalized autoregressive pretraining method which is built on ideas the state-of-the-art autoregressive model of Transformer-XL into pretraining.
- XLNet is designed to learn bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order.
- XLNet is introduced to resolve the issues of neglecting dependency between the masked positions in using BERT. XLNet overcomes the limitations of BERT by using autoregressive formulation.
- XLNet outperforms BERT on 20 tasks, often by a large margin, and achieves state-of-the-art results.

Links for BERT:

Code: https://github.com/google-research/bert
Paper: https://arxiv.org/pdf/1810.04805.pdf

Links for XLNet:
Code: https://github.com/zihangdai/xlnet
Paper: https://arxiv.org/pdf/1906.08237.pdf

Appendix:

- Trends in NLP in the last two years: https://www.analyticsvidhya.com/blog/2019/08/complete-list-important-frameworks-nlp/
- Recent developments in NLP for transfer learning (illustration of several important models): http://jalammar.github.io/illustrated-bert/
- Recent NLP models in Pytorch: https://github.com/huggingface/pytorch-transformers

Project Idea 6: Unsupervised Data Augmentation

(TA: Peijun)

Implement "Unsupervised Data Augmentation" by Xie, Qizhe, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le (2019) and reproduce the results on those six language tasks (you don't need to implement the models for vision tasks).

Description:

- Unsupervised Data Augmentation (UDA) is a model that applies data augmentation to unlabeled data in a semi-supervised learning setting.
- UDA encourages the model predictions to be consistent between an unlabeled example and an augmented unlabeled example.
- Rather than using random noise such as Gaussian noise or dropout noise, UDA uses realistic noise generated by state-of-the-art data augmentation methods.
- UDA makes improvements on six language tasks and outperforms the state-of-the-art model trained on 25,000 labeled examples with only 20 labeled examples.

Links:
Paper: https://arxiv.org/abs/1904.12848
Code: https://github.com/google-research/uda (in Tensorflow)

Data Sets:
IMDb, Yelp-2, Yelp-5, Amazon-2, Amazon-5 and Dbpedia

Project Idea 7: Neural Hawkes model

(TA: Lei Fan)

Implement the Neural Hawkes model for sequential events prediction. Reproduce the results on stack overflow and the retweet dataset.

Description:

- We focus on the problem that: given a series of arrival time and types of action, we predict the next event time and type. Arrival time are generally modeled using point process models. So this project is mathematical.
- Neural Hawkes replace the intensity function of the Hawkes process with a continuous-time LSTM.
- Since we have relatively small public datasets, training doesn't take too much time. CPU is enough.
- In this project, you need to use mask to deal with sequences with various length.
- Grading is based on: your accuracy on action types prediction and rmse for arrival time prediction.

Code:
https://github.com/HMEIatJHU/neurawkes

Paper: http://papers.nips.cc/paper/7252-the-neural-hawkes-process-a-neurally-self-modulating-multivariate-point-process.pdf

Dataset:
https://drive.google.com/drive/u/0/folders/12RLDZTfsR6OF2pFJA10d6YSPFDu5YDRL

Project Idea 8: LSTM, bi-LSTM and Attention model

(TA: Lei Fan)

Implement a simple LSTM, bi-LSTM and Attention model for Yelp Review Polarity. It is a binary classification problem for NLP. You should compare the accuracies for these three models.

Description:

- Sentiment analysis is a popular topic in NLP. This project helps you apply LSTM to solve real world problems. You can start by building a basic LSTM model and gradually go to more complicated models.
- Yelp Review Polarity is a publicly available dataset on Kaggle. It contains customers' reviews and their attitudes, positive or negative. It is a binary classification problem.

Code:
https://github.com/leifanus/nlp-tutorial

Paper:
https://arxiv.org/pdf/1706.03762.pdf

Dataset:
https://www.kaggle.com/irustandi/yelp-review-polarity/version/1#yelp_review_polarity_csv.zip

Project Idea 9: Neural Discrete Representation Learning (VQ-VAE)

(TA: Xiaobo)

Implement a multi-scale hierarchical organization of VQ-VAE, augmented with powerful priors over the latent codes, is able to generate samples with quality that rivals that of state of the art Generative Adversarial Networks on multifaceted datasets such as ImageNet, while not suffering from GAN's known shortcomings such as mode collapse and lack of diversity

Paper:
https://arxiv.org/pdf/1711.00937.pdf
Generating Diverse High-Fidelity Images with VQ-VAE-2

https://arxiv.org/pdf/1906.00446.pdf

Dataset:

Imagenet: http://www.image-net.org/

Project Idea 10: Variational Autoencoders for Collaborative Filtering

(TA: Xiaobo)

Implement a variational autoencoders (vaes) to collaborative filtering for implicit feedback. This nonlinear probabilistic model enables us to go beyond the limited modeling capacity of linear factor models which still largely dominate collaborative filtering research.

Paper:

https://arxiv.org/pdf/1802.05814.pdf

Datasets:

Movie-lens: https://grouplens.org/datasets/movielens/20m/
Netflix Prize (Netflix): https://www.kaggle.com/netflix-inc/netflix-prize-data

Project Idea 11: Distilling the Knowledge in a Neural Network

Implement "Distilling the Knowledge in a Neural Network" by Hinton, Vinyals, and Dean.

Project Idea 12: Faster RCNN

Implement "Faster R-CNN: Towards real-time object detection with regional proposal networks", NIPS, 2015 by Ren et al. in PyTorch. *This is a challenging project.*

Project Idea 13: Fast Fourier Transforms for convolutions

Implement Fast Fourier Transforms (FFT) for convolutions for training convolution networks. See "Fast Training of Convolutional Networks through FFTs" by Mathieu, Henaff, and LeCun (2013) and "Fast Convolutional Nets with fbfft: A GPU Performance Evaluation" by LeCun et al. (2015). This is a challenging project.

Project Idea 14: Image Ranking

Implement a deep learning model for image ranking.
The goal of this project is to introduce you to the computer vision task of image similarity. Like most tasks in this field, it's been aided by the ability of deep networks to extract image features. The task of

image similarity is retrieve a set of n images closest to the query image. One application of this task could involve visual search engine where we provide a query image and want to find an image closest that image in the database. Your task, for this project, will be to implement a simplified version of the pipeline introduced in “Learning Fine-grained Image Similarity with Deep Ranking”.

We strongly encourage you to read this pager - https://static.googleusercontent.com/media/research.google.com/en//pubs/ archive/42945.pdf.\

Here’s another good resource - https://medium.com/@akarshzingade/image-similarity-usingdeep-ranking-c1bd83855978.

Further details on https://courses.engr.illinois.edu/ie534/fa2019/ImageRankingProject.pdf

Project Idea 15: Convolutional Image Captioning (Difficulty: Moderate)

(TA: Jyoti Aneja)

Conventional sequence-to-sequence tasks e.g. machine translation, image captioning etc. are done using LSTMs. LSTMs are inherently sequential in nature. This paper performs image captioning using a completely convolutional approach which results in parallel training over the caption length. Apart from preforming on par with the standard LSTM approach, the convolutional variant results in more diverse captions.
Your tasks:
(a) Re-implement this convolutional approach. Try different (more recent) image encoders and report the improvement.
(b) What happens when instead of standard softmax, you implement a temperature based softmax, Provide a plot of accuracy vs temperature.
(c) The code also has beam search implemented. Compare accuracy and diversity results of best performing beam out of beam size 5 with the accuracy and diversity obtained from the best temperature value in part (b).

Paper: http://openaccess.thecvf.com/content_cvpr_2018/CameraReady/3325.pdf

Datasets: Use the 2014 Train, Val, Test data http://cocodataset.org/#download

Pytorch code: https://github.com/aditya12agd5/convcap.git

Expectations:

- Successful implementation of the convolutional image captioning approach including beam search.
- Plot loss and accuracy curves for best performing model.
- Plot accuracy curves with temperature.
- Report performance on accuracy metrics like BLEU-4, METEOR, CIDEr etc. for at least 2 image encoders different from those mentioned in the paper.
- Visualize results on a webpage in a table with images and corresponding captions for 25 test images.

Project Idea 16: Towards Visual Question Answering Models That Can Read – (Difficulty: High)

(TA: Jyoti Aneja)

Standard Visual Question answering involves machines answering questions, based on the content in an image. Although current models perform decently on this task, they fail miserably when the questions are pertaining to the text in the image.
To this end, a new task called TextVQA was recently introduced. The corresponding dataset has paired image, question, answers and text recognition information provided. The dataset contains a mix of questions which may or may not involve reading the text in the image to answer.
Current best performing model is LoRRA released by Facebook AI Research.
Your Task:

(a) Set up the code repository and understand the how different pieces of the model are being combined
(b) Replicate the result in table 3 of paper i.e: Pythia(I+Q)+LoRRA row.

Paper: https://arxiv.org/pdf/1904.08920.pdf

Datasets: https://textvqa.org/dataset

Pytorch code: https://github.com/facebookresearch/pythia

Expectations:

- A detailed report explaining each component of the LoRRA model
- Successful coding implementation of the LoRRA
- Plot training loss curves vs epoch
- Replicate numbers in table 3 of the provided paper.

References:

[1] Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired image-to-image translation using cycle-consistent adversarial networks." In Proceedings of the IEEE international conference on computer vision, pp. 2223-2232. 2017.

[2] https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/

[3] Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan. "Show and tell: A neural image caption generator." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3156-3164. 2015.

[4] Chen, Xinlei, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO captions: Data collection and evaluation server." arXiv preprint arXiv:1504.00325 (2015).

[5] Plummer, Bryan A., Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models." In Proceedings of the IEEE international conference on computer vision, pp. 2641-2649. 2015.