Course: DD2424 - Information about the project and some project ideas

Teachers and TAs of DD2424 2021

April 1, 2021

It might be daunting, given the amount of material on the web, to try and narrow down the scope and subject matter of your project. In this document we will help you achieve this by suggesting some interesting projects that you could complete and where labelled datasets exist.

1 General guidelines

What recognition, synthesis, manipulation or translation (or some other) task interests you? Is there a network architecture or training algorithm that you would love to investigate in further detail? Is there a way you can explore this interest in a meaningful yet computational feasible way? Or is there a particular application that interests you and can potentially be tackled/solved by deep learning? If you think the answer is yes to any of these questions, then you may have found the topic for your project. In all cases you should study the literature and the plethora of tutorials and blog posts on the web regarding your project idea. Find out the most common and successful deep learning solution to your project topic. If it's an architecture or training algorithm you're investigating then find out what types of problems this network is used to solve and what datasets exist that you can use for your experiments.

You can replicate the results of a published paper, however, you need to do so in a *questioning* manner (what component of the introduced method was crucial to the final performance, can the method be applied to different data than in the original paper, ...). For any project you complete you need to define some relevant questions on the different aspects of the deep learning approach you chose. Conduct meaningful experiments regarding those questions and make and discuss the conclusions that can be made from these experiments. The set of questions can include trying the various techniques taught during the lectures. But you are not limited by the content of the lectures. **We will not accept, though, projects with a focus on re-inforcement learning. This topic is out-of-scope for this course so we will not allow them.**

1.1 Your own specification

You are absolutely free to choose any task or idea you are excited about and a corresponding dataset. But we would advise that you use some paper to help guide your project. You should also consider the size of your dataset relative to your computational resources so it is find to downsample the dataset to make working on it feasible. Be aware that training a "non-classification" network

can be sometimes more challenging. For links and ideas for different datasets please refer to the extra resources section below.

1.2 Project Ideas: For those aiming for a $\leq$ C

Perhaps you are aiming for a somewhat straight-forward implementation of concepts, training and network architectures that were covered in the lectures. You would like to get a feel for 1. how to implement them with a modern deep learning package, 2. how easy is it to replicate results for decent sized datasets (not huge datasets) and 3. how well they work and what is crucial to getting decent performance. Here are a few ideas of projects that would match this ambition level"

- **Basic project**: *Explore training ConvNets from scratch for image classification*. One of the classic tasks of computer vision is image classification. And you have explored this problem within the assignments but only with fully connected networks. However, much of the success deep learning to image classification is due to Convolutional Networks and this project could be a chance to get practical experience writing code to set up the architecture etc and training them. One idea could be to train a deep ConvNet, with an architecture similar to AlexNet or VGGNet, to perform image classification using the Cifar10 dataset. Within this basic project to get a could execution score you would have to explore the benefits to training and generalization of adding (some decent subset of) data-augmentation, dropout, batch normalization, type of optimizer used during training, etc...

  You could potentially bump up the level-of-difficulty with the following:

  1. Train the network to solve ImageNet. ImageNet contains 1000 classes and $\sim$ 1.3 million images. To make things computational feasible given your resources, you will use a 200 classes subset of those images which are also down-sampled from the original size (tiny_imagenet-200.zip). These are the sets used by *Stanford's cs231 course* and are kindly shared by the instructors of that course. Of course, you are free to use the original ImageNet training and test set for your experiments if you have access to computational resources (modern GPUs), however, proceed with caution.
  2. For your ConvNet adopt a more recent architecture (which include various forms of skip-connections) such as ResNet, WideNet, DenseNet, etc....
  3. Do some more sophisticated data-augmentation.

- **Basic project**: *Transfer learning from ImageNet*. In the lectures you have been told that large and deep networks trained on ImageNet can be used as the starting point to solve other computer vision tasks in a process called *Transfer learning*. For this project you would download a pre-trained modern network and then fine-tune it to solve other computer vision tasks. This would involve changing the output layer to suit the new task and then applying fine-tuning (mini-batch gradient descent training). Here you would explore scenarios between the two extremes of freezing the weights for all the layers in the pre-trained network and just learn the final new classification layer and allowing all the parameters of the network to be updated. Here you could explore whether for different datasets and tasks whether the best option to fine tuning varies from between datasets and if there is a pattern.

  Here are some ideas to bump up the level-of-difficulty:

1. Compare the results you achieve with fine-tuning to training a medium sized network from scratch to solve the task.
2. Investigate the correlation between the benefit of fine-tuning and the best number of layers to freeze given the number of training examples.
3. Neural networks suffer from the problem of *catastrophic forgetting*. Say you train a network to perform one task and you then continue training this network to perform another task. If the second round of training is sufficiently long then the network is probably going to have forgotten how to solve the initial task. You could explore this issue.

- **Basic project**: *Train LSTMs to perform text generation* (Generating Sequences With Recurrent Neural Networks by A. Graves, arXiv, 2014) In assignment 4 you trained an RNN to generate passages of text. You may be interested in exploring whether an LSTM could do a better job. The most basic version of the project would be that your input to the LSTM are characters. You could then explore the how different architectures of the LSTMs affect the results. You could also try to apply a ConvNet to the problem and see if it has similar performance to the LSTM etc.

  Here are some ideas to bump up the level-of-difficulty:

  1. Use words as the basic input, as opposed to characters, into the network and use pre-trained word-embeddings - word2vec, BERT, ALBERT or GPT to convert each word to a vector. Note this approach can be harder to implement on less formal text where there may be words that are not in the vocabulary.
  2. Investigate using BERT, ALBERT or GPT-2 embeddings into the model or transformer networks.

1.3 Paper Ideas: Aiming for grade $\geq B$

The following are some suggestions (a drop in the ocean) of papers that could form the basis of your project and we would anticipate that the following papers would be good basis for students aiming to get a grade $\geq B$ and would be feasible to re-implement in the time-frame of the project. Many of the papers below have implementations available online. However, we expect each group to write most of the code themselves. Yes you can refer to online implementations to help clarify implementation details etc. But, one of the goals of the project is to get comfortable using modern deep learning packages so that you can complete your own projects after the course.

But, of course, we are very also very aware that you may not be able to replicate all the experiments on all the datasets (especially the big ones) given the time and computational resources available to you! So get advice from the TAs about what is realistic.

- **Better understanding of convolutional networks**

  - The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks by J. Frankle and M. Carbin published at ICLR, 2019
  - Do Deep Convolutional Nets Really Need to be Deep and Convolutional? by G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson, ICLR, 2017.

- **Batch Normalization and other normalization scheme**

  - How Does Batch Normalization Help Optimization? by S. Santurkar, D. Tsipras, A. Ilyasa and Aleksander Madry, NeurIPS, 2018.
  - EvalNorm: Estimating Batch Normalization Statistics for Evaluation by S. Singh and A. Shrivastava published at ICCV, 2019
  - Norm matters: efficient and accurate normalization schemes in deep networks by E. Hoffer, R. Banner, I. Golan and D. Soudry, NeurIPS, 2018

- **Unpaired translation learning**

  - Colorful Image Colorization by R. Zhang, P. Isola, and A. Efros, ECCV, 2016 (Fun project but I would only recommend it if you are relatively proficient with software engineering. You need to code running so that it can process a lot of image data on your GPU relatively efficiently to get decent results in a sensible time.)

- **Semi-supervised learning**

  - MixMatch: A Holistic Approach to Semi-Supervised Learning, D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver and C. A. Raffel, NeurIPS 2019
  - Unsupervised Representation Learning by Predicting Image Rotations by Spyros Gidaris, Praveer Singh, and Nikos Komodakis, ICLR 2018.
  - Self-Supervised Learning of Pretext-Invariant Representations by I. Misra and L. van der Maaten, CVPR 2020

- **Some crazy data-augmentation technique**

  - mixup: BEYOND EMPIRICAL RISK MINIMIZATION by H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz, ICLR 2018

- **Learning with noisy labels**

  - Symmetric Cross Entropy for Robust Learning With Noisy Labels by Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi and J. Bailey, ICCV 2019

- **Memory and computationally efficient networks**

  - ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices by X. Zhang, X. Zhou, M. Lin and J. Sun, CVPR, 2018
  - Distilling the Knowledge in a Neural Network by Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean, NIPS Deep Learning and Representation Learning Workshop (2015)

- **Analyzing medical imagery**

  - U-Net: Convolutional Networks for Biomedical Image Segmentation by O. Ronneberger, P. Fischer, and T. Brox

- **Auto-encoders**

- Memorization in Overparameterized Autoencoders by A. Radhakrishnan, M. Belkin, C. Uhler, ICML, 2019

- **Improving performance of Convolutional network**

  - Squeeze-and-Excitation Networks by J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, CVPR 2018
  - Attention Augmented Convolutional Networks by I. Bello, B. Zoph, A. Vaswani, J. Shlens, Q. V. Le, ICCV, 2019
  - Drop an Octave: Reducing Spatial Redundancy in Convolutional Neural Networks With Octave Convolution by Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, S. Yan, J. Feng published at ICCV, 2019
  - Information Entropy Based Feature Pooling for Convolutional Neural Networks by W. Wan, J. Chen, T. Li, Y. Huang, J. Tian, C. Yu and Y. Xue published at ICCV, 2019

- **Transformer Networks**

  - I would be very interested in finding out how well a Transformer Network adapted to Assignment 4 would perform and also how much computational effort and resources would be required to get interesting results.

2 Resources

2.1 Publicly available computer vision datasets

Here are some classic visual recognition and classification tasks with a corresponding dataset:

- Face recognition/identification/verification: To uncover/match the identity of a face. Dataset: LFW
- Scene classification: To classify a given image into different scene labels. Dataset: Places
- Semantic segmentation: To classify every pixel of an image into a class. Dataset: MS COCO
- Human detection: To detect the location of all the humans in an image (if any). Datasets: Caltech, Daimler
- Pose estimation: To estimate the location of the joints of a given person. Dataset: MPII
- Video classification: Classify a video into a set of pre-defined classes. Dataset: Youtube Sports
- Depth estimation: Estimate the depth at each pixel from the RGB data. Dataset: NYU
- Gaze estimation: Estimate the eyes gaze of a person based on his image. Dataset: MPII Gaze

The webpages CV Datasets on the web and Yet Another Computer Vision Index To Datasets (YACVID) have a more comprehensive list of available datatsets.

2.2 Resources for Natural language processing

NLP is a domain where variations of RNNs and Transformer networks come into their own. You could potentially use some variation of RNN/transformer network to perform some form of text classification, generation and/or translation. The excellent site spro/practical-pytorch has several links to several tutorials, such as Practical PyTorch: Translation with a Sequence to Sequence Network and Attention or Practical PyTorch: Classifying Names with a Character-Level RNN exercise suggestions. These tutorials and exercise suggestions could definitely form the basis for interesting projects.

2.3 Resources for Speech/Sound

Unfortunately, my hands-on experience in speech recognition and processing is very limited. But I'm very open to finding out more by reading your projects in the area. Here are two Speech Recognition datasets that may be of practical help:

- CMU Robust Speech Recognition Group: Census Database
- LibriSpeech ASR corpus.

Or here is a link to code, a dataset and paper description of how to use a LSTM to *compose* folk music Folk music style modelling using LSTMs. It is unclear to me how long training would take in this case.

3 Popular software packages for deep learning

These frameworks are currently the most popular among academic researchers. (Most take advantage of Nvidia's deep learing libraries so their computational timings do not differ that much.)

- MatConvNet: by Andrea Vedaldi et al. from Oxford University based on `MATLAB`.
- Caffe: created by Yangqing Jia and maintained by BVLC at Berkeley and open-source community. In `C++` and `Cuda` with limited python and MATLAB wrappers.
- PyTorch: Maintained collaboratively by people at Facebook, NYU, ParisTech, Nvidia, ... Written in `python`.
- TensorFlow: The open-source Google deep learning framework. It has both a `C++` and `python` API.
- Deeplearning4j: by SkyMind a San Francisco-based business intelligence and enterprise software firm. Written in `Java` and `Scala`.

If I've missed a popular package, please let me know and I can add it. You are free to choose whichever package you like and feel most comfortable with.

4 Top-tier conferences for inspiration

Here are some top-tier conferences where you can find relevant papers from the fields of computer vision and deep learning:

- Main focus **Computer vision**
  i) ICCV ii) ECCV iii) CVPR iv) BMVC
- Main focus: **Spoken Language Processing**
  i) InterSpeech ii) EuroSpeech
- Main focus: **Natural Language Processing**
  i) InterSpeech ii) EMNLP
- Main focus: **Learning Representations**
  i) ICLR
- Main focus: **Machine Learning**
  i) NeurIPS ii) ICML

Once you have identified a topic of interest, you can search Google Scholar with related keywords on an academic search engine: http://scholar.google.com