DD2424, 2022 - Help defining a project

Teachers and TAs of DD2424 2022

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. In particular we define two categories of projects you can choose from.

One is the set of **default projects** where each group builds the same model to ensure a strong baseline and the group will receive an E grade if they achieve a sufficient level of performance and write a clear and decent report. Within the default project a group can then aim for a higher grade if you complete an extension and/or extensions from the baseline. The increase in grade depends on the level of ambition of the extension, the quality of the implementation and the report. The other project type is the **custom project** where you define your own project and the grade once again depends on the level-of-ambition of the project and the quality of the experimentation and the report.

Ideally we would like students who do not have so much software or deep learning experience to complete a default project, but if your group is relatively experienced with deep learning and/or software engineering then we see no problem with you defining a custom project.

Regardless of which project type you complete, your group will still need to complete a project proposal form.

1 Default Projects

Here is the list of default projects.

1.1 Explore Transfer Learning

The first default project will explore the concept of transfer learning. This is one of the most common use cases of the deep learning - download a pre-trained model and then adapt it your dataset. Please visit the tutorial FINETUNING TORCHVISION MODELS, this gives an overview of how to perform fine-tuning of with a pre-trained ConvNet within PyTorch. Another tutorial is given at FastAI Computer Vision tutorial.

1.1.1 Basic project to get E

To get an E grade you will need to perform the following

- Download a pre-trained modern ConvNet such as: ResNet18, ResNet34,...
- Download the dataset The Oxford-IIIT Pet Dataset
- Replace the final layer of the pre-trained ConvNet to solve the binary classification problem of recognising pictures of Dog Vs Cat. Fine-tune the replaced final layer with the Pet Dataset's training data. Use Adam or NAG optimizer. Without out too much effort you should be able to get very high performance ($\geq 99\%$ test accuracy) on this binary classification task.

  Note you will have to check what spatial size of input your pre-trained network can cope with. The default ResNet architectures have a global average pooling layer, just before the final fully connected output layer, that produces a feature vector of fixed size independent of the spatial extent of the input image. Thus you just have to ensure that the amount of down-sampling implemented during the ResNet does not make the spatial extent of the feature maps disappear for the later layers given the size of your input images. If your images are too small and this will happen then you should re-size them to the smallest acceptable size so that your computational effort is minimized. The normal procedure is to resize the image with one scale factor, to maintain the image's aspect ratio, so its shortest side is re-scaled to the target length.

- Next your goal is to solve the multi-class classification problem of recognising the breed of cat or dog. In this case you have to replace the final layer to have 37 outputs. As this multi-class problem is harder than the binary classification problem you have just solved, you will have to do more work and you should fine-tune more of the network as opposed to just the replaced final layer. You should explore the following issues when fine-tuning:
  - Benefit of fine-tuning more layers (from just the final layer, and then incrementally increasing the number of layers fine-tuned).
  - Different learning rates and/or learning rate schedulers for different layers
  - Benefit of applying data augmentation during training (flip, small rotations, crops, small size scaling)
  - Effect of fine-tuning or not the batch-norm parameters and updating the estimate of the batch mean and standard deviations on the final performance on the new dataset.

After these experiments you should be able to get a final test accuracy of $\geq 95\%$. You do not have to use the Pet Dataset. You are free to use another dataset but it should have comparable or greater difficulty than the Pet Dataset w.r.t. number of classes and size of images.

1.1.2 Potential extensions to go from E $\rightarrow$ D/C

Once you have explored relatively thoroughly the basic project then your group can add extensions to aim for a higher grade. If this higher grade is either a D or C then here are couple of extensions you could apply to investigate if it was possible to improve performance:

- Explore using deeper networks than you used in the basic project. Does the deeper network help? Is it trickier to fine-tune?
- Change the loss you use for the multi-classification problem to say multi-class BCE training or multi-class SVM etc. Does your new loss improve the final test accuracy or speed up convergence etc...
- Just fine-tune the batch norm mean and standard deviation and keep the weights of the layers (except the final layer) frozen and see if you can improve results.
- Add more sophisticated data-augmentations such as including affine transformations and photometric augmentations.

You are, of course, encouraged to come up with your own extensions. But do remember to get them vetted through your project proposal.

1.1.3 Potential extensions to go from E $\rightarrow$ B/A

Once you have explored relatively thoroughly the basic project then your group can add extensions to aim for a higher grade. If this higher grade is either a B or an A, here is an extension suggestion you could apply to investigate if you can apply fine-tuning with fewer labelled training examples and still get good performance:

- Try decreasing the percentage of the labelled training data used during fine-tuning - all the training data, 50%, 10%, 1%. Keep a record of the drop in performance as the percentage of training data used is dropped. In this extension you can explore whether it is possible to use only a small percentage of the training data with their labels plus the rest of the training data without their labels which should be considered as the unlabelled training data set and get the same performance as when you use the full labelled training dataset during fine-tuning. Using both labelled and unlabelled training datasets is known as semi-supervised learning and there has been an explosion of methods and approaches to learn under this regime. The following blog gives a very extensive and readable overview: Learning with not Enough Data Part 1: Semi-Supervised Learning. For your project you are, of course, only expected to explore one approach or paper such as one focusing on
  - Some form of consistency regularization or

- Pseudo-labelling
- etc...

You are, of course, encouraged to come up with your own extensions. But do remember to get them vetted through your project proposal. If you do go for an extension from an E to an A then most of the project report should be devoted to the extension as opposed to the basic project. For the basic assignment you should report the main results and put the more extensive fine-tuning results in the appendix.

1.2 Building and training a modern ConvNet architecture from scratch

The second default project will explore the concept of training a ConvNet from scratch to classify Cifar-10. In the basic project you will retrace the steps of the tutorial How to Develop a CNN From Scratch for CIFAR-10 Photo Classification plus make some simple investigations. Once you have completed the basic network to get a relatively high performance then if you wish to aim for a higher grade you can pursue extensions.

1.2.1 Basic project to get E

The basic project will involve constructing a baseline network architecture based on the VGG network to solve the Cifar-10 dataset. The baseline network you will construct will have three VGG blocks as described in How to Develop a CNN From Scratch for CIFAR-10 Photo Classification. You can use whichever deep learning software package you like, not specifically Keras, the tutorial is mainly a guideline for the specific architecture to use and the expected performance you should achieve. Initially, train your network with the optimizer SGD + momentum and no regularization. With this network and training you should be able to achieve results $\sim 73\%$.

Next you will explore adding different types of regularization independently to the basic network:

1. Dropout regularly applied throughout the network.
   Target performance: **Baseline + Dropout**: $\sim 83\%$
2. Weight decay a.k.a L2 regularization.
   Target performance: **Baseline + Weight Decay**: $\sim 72\%$
3. Data-augmentation - horizontal flipping + $x-$ and $y-$ translation shifts.
   Target performance: **Baseline + Data Augmentation**: $\sim 84\%$

Then you should train the network with multiple regularization strategies combined to increase performance plus the addition of batch normalization. As you are applying more regularization, you might need to train for longer but the use of batch normalization may allow you to use a higher learning rate. The final training regime should include higher levels of dropout, data augmentation and batch normalization. The target performance of this network but training set-up is $\sim 88\%$.

All these stages are explicitly described in the tutorial. Thus you need also need to perform some explorations not so demarcated in the tutorial. You should try and explore each of the following for the final set-up

- The tutorial normalizes the input data by ensuring it is between 0 and 1. See if normalizing the data to have zero mean and standard deviation 1 (as in the programming assignments) has any effect.
- Replace the SGD + momentum optimizer with Adam. Does this lead to better performance and/or faster convergence?
- Try different learning rate schedulers such as learning rate warm-up + cosine annealing, step decay= or cosine annealing with re-starts and see how is helps/affects training.

- It is not clear in which order Dropout and Batch Norm should be performed. In the tutorial it is BatchNorm then dropout. Check if changing the order to Batch Norm then Dropout has an effect on performance. Also check if the Dropout and Batch Norm are complementary ie having both Dropout and Batch Norm in the network is better or worse than having a network that just has one of these regularization techniques.

1.2.2 Potential extensions to go from E $\rightarrow$ D/C

Once you have explored relatively thoroughly the basic project then your group can add extensions to aim for a higher grade. If this higher grade is either a D or C then here are some extensions you could apply to investigate if it was possible to improve performance. You will probably need to investigate a couple to get up to an C:

- Make the VGG network architecture you have constructed in the basic assignment look more like a ResNet with the following steps:

  - Replace the large fully connected layer + final fc output layer with a *Global Average Pooling* layer + final fc output layer
  - Replace the each MaxPooling layer with a convolution layer applied with stride two.
  - Add a skip connections within the network. Look carefully at the ResNet architecture to see faithful ways to do this.

  Set up this new architecture and see if training becomes easier and/or if better results can be achieved. If there is no difference perhaps make the network deeper with a fourth block and compare the results of the two architectures with this deeper network.

- Try more extensive data-augmentations (more geometric data-augmentations: affine transformations, scaling and rotation and/or photo-metric augmentations.) and see if this gives a performance boost.

- Replace Batch Norm with Layer Norm or Instance Norm or Group Norm and see if you can maintain performance levels with the simpler normalization practice.

1.2.3 Potential extensions to go from E $\rightarrow$ B/A

Once you have explored relatively thoroughly the basic project then your group can add extensions to aim for a higher grade. If this higher grade is either a B or an A, here are suggested extensions you could to investigate:

- Train a ResNet architecture from scratch with more bells and whistles that you can find in the literature to get a final test accuracy ≥ 90 on Cifar-10 (within a reasonable training time) and perform training on Cifar-100 to see if the same training and approach can produce good results on a dataset with more classes.

- Speed up training by *quickly and efficiently finding the core set* of examples actually needed for training. There are many ways to do this, but one relatively straightforward and effective

general approach is described in SELECTION VIA PROXY: EFFICIENT DATA SELECTION FOR DEEP LEARNING, Cody Coleman et al., ICLR 2020. The high-level idea is to train a ConvNet, which works well but perhaps not amazingly but can be trained relatively quickly, on all the labelled training data. Use the trained *simple ConvNet* to identify the training examples most useful for training - for example keep the training examples whose outputs have highest entropy. This set is termed the core-set, see section 2.2 of the paper for more details. As usually the training set has a lot of redundancy this core-set may correspond to approximately up to 50% of the original data but obviously this will vary from dataset to dataset etc. If you then train the big complicated network on just the core-set then it is often possible to achieve a final test performance similar to training on all the data but you can train the network trains much more quickly. You can use the large VGG net you trained as your "complicated network" and either a smaller VGG network or the big VGG network not trained for so long as your simple network. Then you can explore finding the core-set with the entropy based measure or other measures (if you have the time or patience to implement them) and then train the large network for a long time on just the core set.

- Contaminate the labels of your training data with noisy labels and investigate how it corrupts training and explore an option in the literature to train a network in the presence of noisy labels. An interesting dataset that has pre-defined label noise but is not too big to perform reasonable computations is Imagenette There is a lot of work in this area. Please check out the Custom Projects to see links to relevant papers in the literature.

You are, of course, very welcome to come up with your own extensions. But do remember to get them vetted through your project proposal. If you do go for an extension from an E to an A then most of the project report should be devoted to the extension as opposed to the basic project. For the basic assignment you should report the main results and put the more extensive fine-tuning results in the appendix.

1.3 NLP project

In this default project you should upgrade Assignment 4 from a simple vanilla RNN to a deeper LSTM. As a reference and some tips of implementation please check out Shakespeare Text Generation (using RNN LSTM)

1.3.1 Basic project to get D

For the basic assignment you should select a distinctive set of data to train on such as:

- The poems of Emily Dickinson: Project Gutenberg's Poems: Three Series, Complete, by Emily Dickinson
- The plays of Shakespeare, a subset of the plays is available here shakespeare.txt.
- Or perhaps don't focus on literature but a programming language instead!

Project Gutenberg is a good source for books that can be downloaded.

Also in this upgrade you should construct training, validation and test sets so that performance can be measured more quantitatively. Besides prediction loss you can also think of other metrics to measure the quality of generated text such as the percentage of correctly spelt words generated. To complete the basic assignment you should:

- Train an RNN baseline on the dataset you use and compare to at least to both a one and two layer LSTM both qualitatively and quantitatively.
- Generate diverse sequences from the training data to construct the training batches.
- Investigate how increasing the number of the nodes of the hidden state increases or decreases performance.
- During text generation you should scale the output probabilities with a temperature to generate more or less predictable passages of text. Also implement and investigation *Nucleus Sampling* as described in The Curious Case of Neural Text Degeneration by Ari Holtzman et al., ICLR 2020.

1.3.2 Potential extensions to go from D $\rightarrow$ C

Once you have explored relatively thoroughly the basic project then your group can add extensions to aim for a higher grade. If this higher grade is either a D or C then here are couple of extensions you could apply to investigate if it was possible to improve performance:

- Compare the effect of GRU Vs LSTM layers
- More thoroughly investigate if more depth helps performance. If you begin to overfit during training add dropout layers and also add layer normalization to help with gradient flow.

You are, of course, encouraged to come up with your own extensions. But do remember to get them vetted through your project proposal.

1.3.3 Potential extensions to go from D → A/B

Once you have explored relatively thoroughly the basic project then your group can add extensions to aim for a higher grade. If this higher grade is either a B or an A, here are suggested extensions you could to investigate:

- Use words as the basic entry in the network and use a standard word embedding such as word2vec etc or Glove.
- Try data-augmentation methods for text such as back-translation - translate a passage of text into one language then translate it back into English - to augment your basic training data. Another option is to replace words with their synonyms.
- When generating text use a beam search instead and see how this improves qualitatively the results.
- Use some of the ideas in VISUALIZING AND UNDERSTANDING RECURRENT NETWORKS by A. Karpathy, J. Johnson, and L. Fei-Fei, arXiv 2015 to help visualize and understand what your LSTM has learnt.

You are, of course, encouraged to come up with your own extensions. But do remember to get them vetted through your project proposal.

2 Custom Final Project

2.1 General guidelines for choosing a project topic

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.**

2.2 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 (from a reputable source) to help guide your project. You should also consider the size of your dataset relative to your computational resources so it is fine to downsample the dataset to make working on it feasible. For links and ideas for different datasets please refer to the extra resources section below.

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

We assume most project groups completing a custom project are aiming for a grade greater than $\geq B$. If this is not the case please review the default projects and the suggested extensions to get an idea of the level we expect for grades E-C.

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 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

- **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 / Contrastive 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**

  - Train a VIT like network on Cifar-10/Cifar-100 and see how it goes! At the moment ViT type models cannot be trained from scratch on small images and relatively small datasets to give similar performance as ConvNets trained from scratch on the same datasets. But this exercise would be more about getting into the details of ViT architecture.
  - In 2021 there was a flurry of work investigating if the attention layers in ViT are strictly necessary. In particular they replace the attention layer with some other operation (without the quadratic complexity) to allow communication between the representations of different patches. Here are two such papers:
    - MLP-Mixer: An all-MLP Architecture for Vision by I. Tolstikhin et al., NeurIPs 2021.
    - Do You Even Need Attention? A Stack of Feed-Forward Layers Does Surprisingly Well on ImageNet by Luke Melas-Kyriazi, arxiv 2021

    One can interpret the architectures described in these papers as very particular forms of ConvNet.
  - In a similar vein to the prior papers there is the extension to this work is the ConvMixer described in Patches Are All You Need? by A. Trockman and J. Zico Kolter. The architecture can be trained on Cifar-10 from scratch more easily to have good performance than the MLP-Mixer. It might also be fun to investigate using a pre-trained version of this architecture to perform semantic segmentation as no downsampling, apart from the patch extraction at the start, is performed in the network.

3 Resources

3.1 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.

3.2 Resources for Natural language processing

NLP is a domain where variations of RNNs and Transformer networks come into their own. The site Hugging Face has many pre-trained models available for downloading. 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. The Stanford course CS224n: Natural Language Processing with Deep Learning is potentially also another great resource and source of project ideas of a more sophisticated and ambitious nature.

3.3 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.

4 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.)

- 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.
- JAX: This is NumPy with automatic differentiation which you can run on CPU, GPU and TPU. It also has just-in-time compilation. It is maintained by Google. Flax is the neural network library for JAX and it is an open source project maintained by Google Research.
- 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.

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.

5 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

References