Deep Learning: Towards Deeper Understanding **8 May, 2018**

Project 3: Final

Instructor: Yuan Yao *Due: 22 May, 23:59, 2018*

Requirement and Datasets

This project as a warm-up aims to explore feature extractions using existing networks, such as pre-trained deep neural networks and scattering nets, in image classifications with traditional machine learning methods.

1. Pick up ONE (or more if you like) favourite dataset below to work. If you would like to work on a different problem outside the candidates we proposed, please email course instructor about your proposal. Some challenges marked by $*$ is optional with bonus credits.

2. Team work: we encourage you to form small team, up to FOUR persons per group, to work on the same problem. Each team just submit ONE report, *with a clear remark on each person's contribution.* The report can be in the format of either Python (Jupyter) Notebooks with a detailed documentation (preferred format), a *technical report within 8 pages*, e.g. NIPS conference style

   `https://nips.cc/Conferences/2016/PaperInformation/StyleFiles`

   or of a *poster*, e.g.

   `https://github.com/yuany-pku/2017_math6380/blob/master/project1/DongLoXia_poster.pptx`

3. In the report, show your proposed scientific questions to explore and main results with a careful analysis supporting the results toward answering your problems. Remember: scientific analysis and reasoning are more important than merely the performance tables. Separate source codes may be submitted through email as a zip file, GitHub link, or as an appendix if it is not large.

4. Submit your report by email or paper version no later than the deadline, to the following address (deeplearning.math@gmail.com) with Title: Math 6380O: Project 3.

1 Reinforcement Learning for Image Classification with Recurrent Attention Models

This task basically required you to reproduce some key result of Recurrent Attention Models, proposed by Mnih et al. (2014). (See `https://arxiv.org/abs/1406.6247`)

1.1 Cluttered MINIST

As you've known, the original MNIST dataset is a canonical dataset used for illustration of deep learning, where a simple multi-layer perceptron could get a very high accuracy. Therefore the original MNIST is augmented with additional noise and distortion in order to make the problem more challenging and closer towards real-world problems.

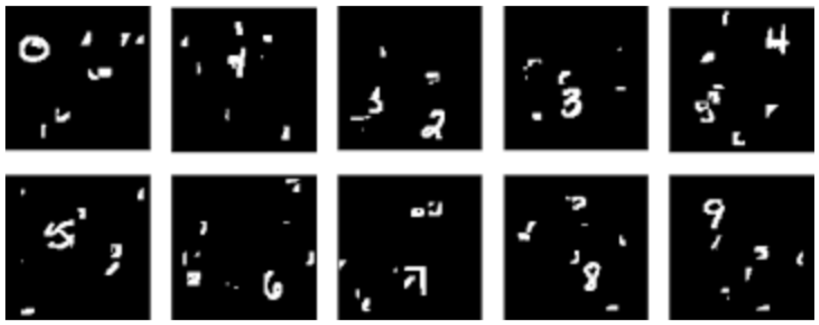

Figure 1: Cluttered MNIST data sample.

The dataset can be downloaded from `https://github.com/deepmind/mnist-cluttered`.

1.2 Raphael Drawings

For those who are interested in exploring authentication of Raphael drawings, you are encouraged to use the Raphael dataset:

`https://drive.google.com/folderview?id=0B-yDtwSjhaSCZ2FqN3AxQ3NJNTA&usp=sharing`

that will be discussed in more detail later.

1.3 Recurrent Attention Models (RAMs)

The efficiency of human eyes when looking at images lies in attentions – human do not process all the input information but instead use attention to select partial and important information for perception. This is particularly important for computer vision when the input images are too big to get fully fed into a convolutional neural networks. To overcome this hurdle, the general idea of

RAM is to model human attention using recurrent neural networks by dynamically restricting on interesting parts of the image where one can get most of information, followed by reinforcement learning with rewards toward minimizing misclassification errors.

You're suggested to implement the following networks and train the networks on the above Cluttered MINIST dataset.

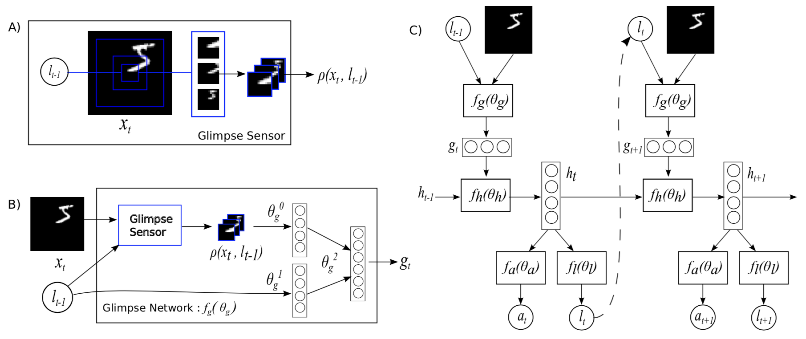

Figure 2: Diagram of RAM.

- Glimpse Sensor: Glimpse Sensor takes a full-sized image and a location, outputs the *retina-like* representation $\rho(x_t, l_{t-1})$ of the image x_t around the given location l_{t-1}, which contains multiple resolution patches.

- Glimpse Network: takes as the inputs the retina representation $\rho(x_t, l_{t-1})$ and glimpse location l_{t-1}, and maps them into a hidden space using independent linear layers parameterized by θ_g^0 and θ_g^1 respectively using rectified units followed by another linear layer θ_g^2 to combine the information from both components. Finally it outputs a glimpse representation g_t.

- Recurrent Neural Network (RNN): Overall, the model is an RNN. The core network takes as the input the glimpse representation g_t at each step and history internal state h_{t-1}, then outputs a transition to a new state h_t, which is then mapped to action a_t by an action network $f_a(\theta_a)$ and a new location l_t by a location network $f_l(\theta_t)$. The location is to give an attention at next step, while the action, for image classification, gives a prediction based on current informations. The prediction result, then, is used to generate the reward point, which is used to train these networks using Reinforcement Learning.

- Loss Function: The reward could be based on classification accuracy (e.g. in Minh (2018) $r_t = 1$ for correct classification and $r_t = 0$ otherwise). In reinforcement learning, the loss can be finite-sum reward (in Minh (2018)) or discounted infinite reward. Cross-entropy loss for prediction at each time step is an alternative choice (which is not emphasized in the origin paper but added in the implementation given by Kevin below). It's also interested to figure out the difference function between these two loss and whether it's a good idea to use their combination.

To evaluate your results, it is expected to see improved misclassification error compared against feedforward CNN without RAMs. Besides, you're encouraged to visualize the glimpse to see how the attention works for classification.

1.4 More Reference

- A PyTorch implementation of RAM by Kevin can be found here:
  `https://github.com/kevinzakka/recurrent-visual-attention`.
- For reinforcement Learning, David Silver's course web could be found here
  `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html`
  and Ruslan Satakhutdinov's course web at CMU is
  `http://www.cs.cmu.edu/~rsalakhu/10703/`

2 Generating Images via Generative Models

In this project, you are required to train a generative model with given dataset to generate new images.

- The generative models include, but not limited to, the models mentioned in 2.1.
- It is suggested to use datasets in 2.2.
- It is recommended to have some analyses and discussion on new images with the trained generative model. You may use some evaluation in 2.3.
- * For those who made serious efforts to explore or reproduce the results in Rie Johnson and Tong Zhang (2018), `https://arxiv.org/abs/1801.06309`, additional credits will be given. You could compare with the state-of-art result WGAN-GP(2017), `https://arxiv.org/abs/1704.00028` and original GAN with/without log-d trick. Moreover, you may try the same strategy in other fancy GANs, e.g. ACGAN (`https://arxiv.org/abs/1610.09585`), CycleGAN (`https://arxiv.org/abs/1703.10593`), etc.

2.1 Variational Auto-encoder (VAE) and Generative Adversarial Network (GAN)

Two popular generative models were introduced in class. You could choose one of them, or both to make a comparison. You could also train the model without or with labels.

2.2 Datasets

2.2.1 MNIST

Yann LeCun's website contains original MNIST dataset of 60,000 training images and 10,000 test images. `http://yann.lecun.com/exdb/mnist/`

There are various ways to download and parse MNIST files. For example, Python users may refer to the following website: `https://github.com/datapythonista/mnist` or MXNET tutorial on mnist `https://mxnet.incubator.apache.org/tutorials/python/mnist.html`

2.2.2 Fashion-MNIST

Zalando's Fashion-MNIST dataset of 60,000 training images and 10,000 test images, of size 28-by-28 in grayscale. `https://github.com/zalandoresearch/fashion-mnist`

As a reference, here is Jason Wu, Peng Xu, and Nayeon Lee's exploration on the dataset in project 1: `https://deeplearning-math.github.io/slides/Project1_WuXuLee.pdf`

2.2.3 CIFAR-10

The Cifar10 dataset consists of 60,000 color images of size 32x32x3 in 10 classes, with 6000 images per class. It can be found at `https://www.cs.toronto.edu/~kriz/cifar.html`

2.3 Evaluation

On one hand, you could demonstrate some images generated by the trained model, then discuss both the pros (good images) and cons (flawed images) and analyse possible reasons behind what you have seen.

On the other hand, you may compare different methods using some quantitative measurements, e.g. *Inception Score* and *Diversity Score* mentioned in class.

- The *Inception Score* was proposed by Salimans et al. (`http://arxiv.org/abs/1606.03498`) and has been widely adopted since. The inception score uses a pre-trained neural network classifier to capture to two desirable properties of generated samples: highly classifiable and diverse with respect to class labelis. It is defined as

$$\exp\left\{\mathbb{E}_x KL(P(y|x)||P(y))\right\}$$

  which measures the quality of generated images, i.e. high-quality images should lead to high confidence in classification whence high inception score. For instance, see the following reference: and `http://arxiv.org/abs/1801.01973`.

- The inception score fails to capture mode collapse inside a class: the inception score of a model that generates the same image for a class and the inception score of a model that is able to capture diversity inside a class are the same. The *Diversity Score* suggested by Tong ZHANG is defined as

$$\exp\{\mathbb{E}_x KL(P(y)||P_*(y))\}$$

  which measures the diversity of generated images such that the score becomes large (approaching to 1) when generated class distribution mimics real class distribution. Another popular measure of diversity is the multiscale structural similarity (MS-SSIM) suggested by Wang et al. (`http://www.cns.nyu.edu/~zwang/files/papers/msssim.pdf`) and introduced to GAN by Odena et al. (`https://arxiv.org/abs/1610.09585`). You may choose these measures to inspect the diversity of generated images quantitatively.

3 Nexperia Predictive Maintenance

Refer to the introduction by Mr. Gijs Bruining:

`https://github.com/deeplearning-math/deeplearning-math.github.io/blob/master/slides/NexperiaContest.pdf`

Kaggle in-class contests are launched at the following website:

`https://www.kaggle.com/c/nexperia-predictive-maintenance`

where three contests are available, depending the data to use for predictions. The number of days before the predictive dates is called the *observation window* (OW) and the number of predictive dates is called the *predictive window*. As a warm-up, the mini-contest is to exploit 2 days observation window (OW=2) with prediction window (PW=1). Two additional full-contests exploits different combinations of OW (=1,2,4,8,16, all in as features) and PW (1 or 2 for two full-contests, respectively).

4 From Project 2: Reproducible Training and Generalizations of CNNs

The following best award paper in ICLR 2017,

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, Understanding deep learning requires rethinking generalization. `https://arxiv.org/abs/1611.03530`

received lots of attention recently. Reproducibility is indispensable for good research. Can you reproduce some of their key experiments by yourself? The following are for examples.

1. Achieve ZERO training error in standard and randomized experiments. As shown in Figure 3, you need to train some CNNs (e.g. ResNet, over-parametric) with Cifar10 dataset, where the labels are true or randomly permuted, and the pixels are original or random (shuffled, noise, etc.), toward zero training error (misclassification error) as epochs grow. During the training, you might turn

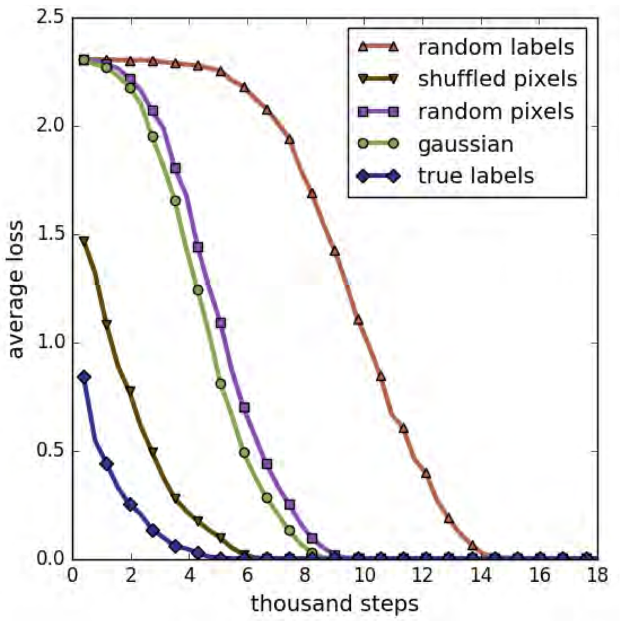

Figure 3: Overparametric models achieve zero *training error* (or near zero *training loss*) as SGD epochs grow, in standard and randomized experiments.

on and off various regularization methods to see the effects. If you use loss functions such as cross-entropy or hinge, you may also plots the training loss with respect to the epochs.

2. Non-overfitting of test error and overfitting of test loss when model complexity grows. Train several CNNs (ResNet) of different number of parameters, stop your SGD at certain large enough epochs (e.g. 1000) or zero *training error (misclassification)* is reached. Then compare the *test (validation) error* or *test loss* as model complexity grows to see if you observe similar phenomenon in Figure 4: when *training error* becomes zero, *test error* (misclassification) does not overfit but *test loss* (e.g. cross-entropy, exponential) shows overfitting as model complexity grows. This is for reproducing experiments in the following paper:

Tomaso Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Biox, J. Hidary, and H. Mhaskar. Theory of Deep Learning III: the non-overfitting puzzle. Jan 30, 2018. `http://cbmm.mit.edu/publications/theory-deep-learning-iii-explaining-non-overfitting-puzzle`

3. Can you give an analysis on what might be the reasons for the phenomena you observed?

The Cifar10 dataset consists of 60,000 color images of size 32x32x3 in 10 classes, with 6000 images per class. It can be found at

`https://www.cs.toronto.edu/~kriz/cifar.html`

Attention: training CNNs with such a dataset is time-consuming, so GPU is usually adopted. If you would like an easier dataset without GPUs, perhaps use MNIST or Fashion-MNIST (introduced below).

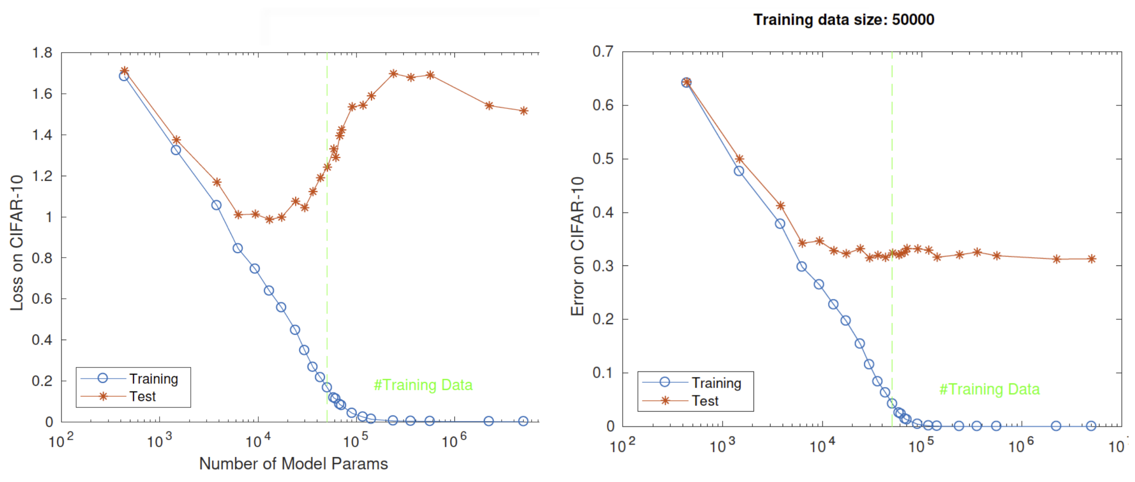

Figure 4: When *training error* becomes zero, *test error* (misclassification) does not increase (resistance to overfitting) but *test loss* (cross-entropy/hinge) increases showing overfitting as model complexity grows.

4.1 Fashion-MNIST dataset

Zalando's Fashion-MNIST dataset of 60,000 training images and 10,000 test images, of size 28-by-28 in grayscale.

`https://github.com/zalandoresearch/fashion-mnist`

As a reference, here is Jason Wu, Peng Xu, and Nayeon Lee's exploration on the dataset in project 1:

`https://deeplearning-math.github.io/slides/Project1_WuXuLee.pdf`

5 From Project 2: Image Captioning by Combined CNN/RNN/LSTM

In this project, you're required to implement a RNN to do image captioning. Your work may include the following parts, but not limited to,

- Implement a CNN structure to do feature selection. You may do this by transfer learning, like using Inception, ResNet, etc.
- Implement a (e.g. single hidden layer) fully connected network to do word embedding.
- Implement a RNN structure to do image caption. You may use select one of network structure, like LSTM, BiLSTM, LSTM with Attention, etc.
- Train your network and tune the parameters. Select the best model on validation set.
- Show the caption ability of your model visually. Evaluate your model by BLEU (bilingual evaluation understudy) score on test set.

5.1 Dataset: Flickr8K

You could download Filckr8K dataset, which includes 8,000 images and 5 captions for each, via the following links. `https://forms.illinois.edu/sec/1713398`

The Flickr8K dataset is provided by flicker, an image- and video-hosting website. It's a relatively small dataset in image captioning community. Perhaps it's still too big for CPU computations. If you don't have access to GPU resources, try using dimension reduction on image features and using pre-trained word embedding to help you work this project on your own CPU.

6 Continued Challenges from Project 1

In project 1, the basic challenges are

- Feature extraction by scattering net with known invariants;
- Feature extraction by pre-trained deep neural networks, e.g. VGG19, and resnet18, etc.;
- Visualize these features using classical unsupervised learning methods, e.g. PCA/MDS, Manifold Learning, t-SNE, etc.;
- Image classifications using traditional supervised learning methods based on the features extracted, e.g. LDA, logistic regression, SVM, random forests, etc.;
- Train the last layer or fine-tune the deep neural networks in your choice;
- Compare the results you obtained and give your own analysis on explaining the phenomena.

You may continue to improve your previous work. Below are some candidate datasets.

6.1 MNIST dataset – a Warmup

Yann LeCun's website contains original MNIST dataset of 60,000 training images and 10,000 test images.

`http://yann.lecun.com/exdb/mnist/`

There are various ways to download and parse MNIST files. For example, Python users may refer to the following website:

`https://github.com/datapythonista/mnist`

or MXNET tutorial on mnist

`https://mxnet.incubator.apache.org/tutorials/python/mnist.html`

6.2 Identification of Raphael's paintings from the forgeries

The following data, provided by Prof. Yang WANG from HKUST,

`https://drive.google.com/folderview?id=0B-yDtwSjhaSCZ2FqN3AxQ3NJNTA&usp=sharing`

contains a 28 digital paintings of Raphael or forgeries. Note that there are both jpeg and tiff files, so be careful with the bit depth in digitization. The following file

`https://docs.google.com/document/d/1tMaaSIrYwNFZZ2cEJdx1DfFscIfERd5Dp2U7K1ekjTI/edit`

contains the labels of such paintings, which are

1 Maybe Raphael - Disputed

2 Raphael

3 Raphael

4 Raphael

5 Raphael

6 Raphael

7 Maybe Raphael - Disputed

8 Raphael

9 Raphael

10 Maybe Raphael - Disputed

11 Not Raphael

12 Not Raphael

13 Not Raphael

14 Not Raphael

15 Not Raphael

16 Not Raphael

17 Not Raphael

18 Not Raphael

19 Not Raphael

20 My Drawing (Raphael?)

21 Raphael

22 Raphael

23 Maybe Raphael - Disputed

24 Raphael

25 Maybe Raphael - Disputed

26 Maybe Raphael - Disputed

27 Raphael

28 Raphael

There are some pictures whose names are ended with alphabet like A's, which are irrelevant for the project.

The challenge of Raphael dataset is: can you exploit the known Raphael vs. Not Raphael data to predict the identity of those 6 disputed paintings (maybe Raphael)? Textures in these drawings may disclose the behaviour movements of artist in his work. One preliminary study in this project can be: *take all the known Raphael and Non-Raphael drawings and use leave-one-out test to predict the identity of the left out image; you may break the images into many small patches and use the known identity as its class.*

The following student poster report seems a good exploration

`https://github.com/yuany-pku/2017_CSIC5011/blob/master/project3/05.GuHuangSun_poster.pdf`

The following paper by Haixia Liu, Raymond Chan, and me studies Van Gogh's paintings which might be a reference for you:

`http://dx.doi.org/10.1016/j.acha.2015.11.005`

In project 1, some explorations can be found here for your reference:

1) Jianhui ZHANG, Hongming ZHANG, Weizhi ZHU, and Min FAN: `https://deeplearning-math.github.io/slides/Project1_ZhangZhangZhuFan.pdf`,

2) Wei HU, Yuqi ZHAO, Rougang YE, and Ruijian HAN: `https://deeplearning-math.github.io/slides/Project1_HuZhaoYeHan.pdf`.

Moreover, the following report by Shun ZHANG from Fudan University presents a comparison with Neural Style features:

3) `https://www.dropbox.com/s/ccver43xxvo14is/ZHANG.Shun_essay.pdf?dl=0`.