EEP 596 Homework 3 (100%) due: 11/09 11:59 pm

Submission Instructions

You will need to submit the following materials:

1. **Questions and Answers** in PDF format. Remember to put your name in your report.
2. A folder containing all ipynb files with your results for the the problems. Please remember to keep all results and logs in the submitted ipynb files to get full credit.

All submitted files should be put into one single zip file named as HW#_xxx.zip, e.g. HW3_George_Clooney.zip**,** including all ipynb files with answers (both results and discussions), and the **Questions and Answers** report.

Problem 1: Image Classification on Imbalanced Dataset (25%)

In the existing visual recognition setting, the training data and testing data are both balanced under a closed-world setting, e.g., the ImageNet dataset. However, this setting is not a good proxy for the real-world scenario. For example, it is never possible for ecologists to gather balanced wildlife datasets because animal distribution is imbalanced. This imbalanced data distribution in the training set may largely degrade the performance of the machine learning or deep learning based method.

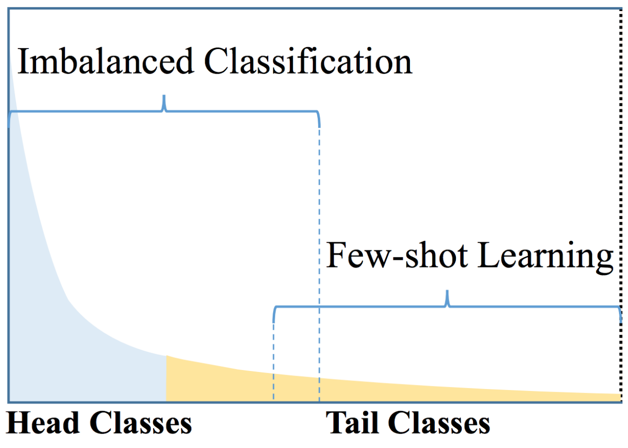


Fig.1 Our task of long-tailed recognition must learn from long-tail distributed training data and deal with imbalanced classification and few-shot learning over the entire spectrum.

Therefore, to faithfully reflect these aspects, we have to carefully study **Long-Tailed Recognition (LTR)** arising in natural data settings. A practical system shall be able to classify among a few common and many rare categories, to generalize the concept of a single category from only a few known instances. We define LTR as learning from **long-tail distributed**

training data and evaluating the classification accuracy over **a balanced test set** which includes head and tail in a continuous spectrum (Fig. 1). Sometimes the training data can be so few, they cannot be used for training, and we have to resort to **few shot learning** methods.

Dataset The original version of CIFAR-10 and CIFAR-100 contains 50,000 training images and 10,000 testing images of size 32×32 with 10 and 100 classes, respectively. The original datasets and detailed descriptions can be found here: https://www.cs.toronto.edu/~kriz/cifar.html. For this question, we will use the CIFAR-50-LT dataset specifically collected from the CIFAR-100 dataset.

- Q1: Download dataset to your Google Drive from https://drive.google.com/drive/folders/1WGUKBP5Eta9DAltK1WtvRbX43iwP08DU?usp=sharing and unzip the files. Please follow the instructions in the notebook to load/print the labels as well as visualize some images.

Method

- Q2 (4%): Train the CNN
  a. Use the CNN in HW2 to train the model on the balanced CIFAR50 dataset and train the CNN on the balanced CIFAR50 training set. **Evaluate and report** the classification accuracies on the testing set. Note: You can use any network configurations you implemented in HW2.
  b. Use the same CNN in HW2 to train the model on the **imbalanced** CIFAR50 dataset and Train the CNN on the imbalanced CIFAR50 training set. **Evaluate and report** the classification accuracies on the testing set.

- Q3 (21%): Implement tricks on your CNN for imbalanced dataset
  a. Before starting this question, please refer to the **paper on canvas** for this homework: **Bag of tricks for long-tailed visual recognition with deep convolutional neural networks**. Select **three** tricks to implement on the imbalanced CIFAR50 training, ex.,
    (1) Re-sampling
    (2) Re-weighting
    (3) Mixed-up
  b. **Evaluate and report** the classification performance on the CIFAR50 testing set.

Problem 2: Transfer Learning and Domain Adaptation (25%)

The goal of **domain adaptation** is to transfer the knowledge of a model to a different but related data distribution. The model is trained on a source dataset and applied to a target dataset (usually unlabeled). For Problem 2, the model will be trained on regular MNIST images, but we want to get good performance on MNIST with random color (without any labels).

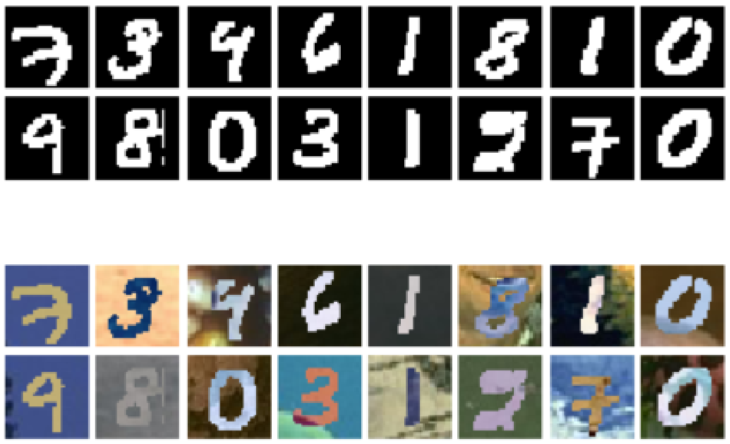


Fig.2 Above: MNIST dataset (original), Bottom: MNIST-M dataset (with random colors, without labels)

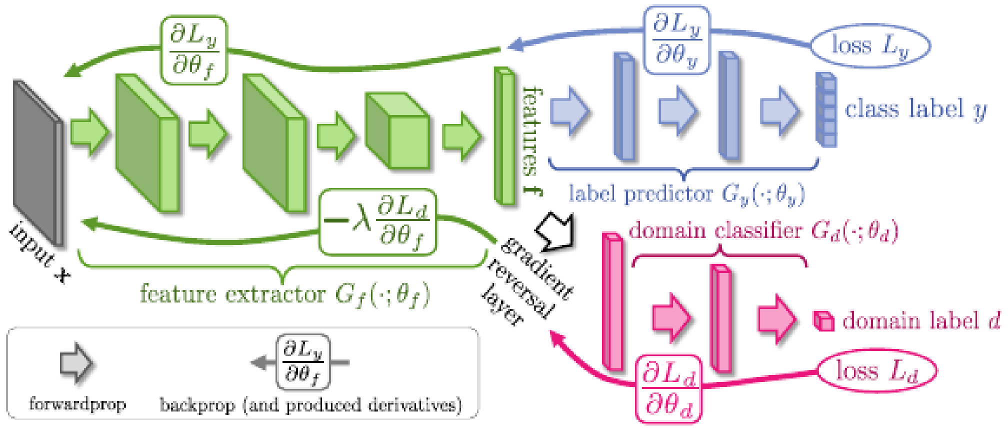
Problem Statement Given a labelled source domain (MNIST) and an unlabelled target domain (MNIST-M). We would like to train a classifier or a predictor which would give accurate predictions on the target domain.

Assumptions Probability distribution of source domain is not equal to the probability distribution of target domain. The conditional probability distribution of the labels given an instance from the source domain is equal to the conditional probability distribution of the labels given an instance from the target domain. Source dataset is labelled. Target dataset is unlabelled.

Approach Here, we adopt the DABP method mentioned in the paper "Unsupervised Domain Adaptation by Backpropagation".

- Feature Extractor (green): This is a neural network that will learn to perform the transformation on the source and target distribution.
- Label Classifier (blue): This is a neural network that will learn to perform the classification on the transformed source distribution. Since, the source domain is labelled.
- Domain Classifier (red): This is a neural network that will predict whether the output of the Feature Extractor is from the source distribution or the target distribution.

By using the above three components, the Feature Extractor will learn to produce discriminative and domain-invariant features.



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
input $\mathbf{x}$
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
$\frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
backprop (and produced derivatives)

Fig3. Unsupervised Domain Adaptation by Backpropagation

(a) Follow the instructions in the notebook to download the pre-processed MNIST and MNIST-M dataset and visualize some examples.

(b) The details of the implementation are also introduced in the notebook. Please read it carefully before working on this problem. The implementation of the DABP includes the following components (please follow the provided sample codes):
  (i) MNIST and MNITS-M DataLoader
  (ii) Feature Extractor
  (iii) Label and Domain Classifier
  (iv) Gradient Reversal Layer

(c) Train the DABP model by running `main.py` and answer the following questions.

- Q2 (10%): **Perform 3** experiments on training and report your source and target accuracy in the tables below. (Your result is the average of the Target Accs. based on 3 experiments)

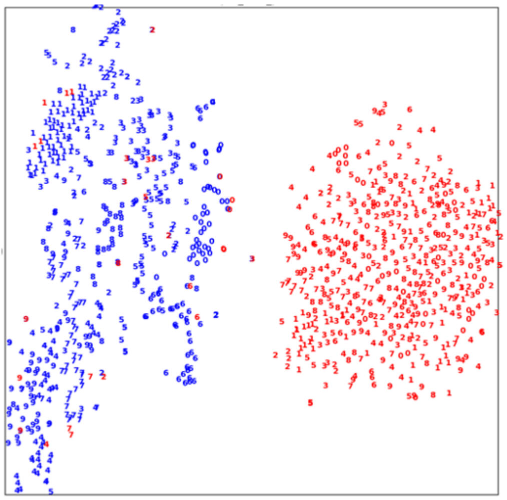
source_only

	Test1	Test2	Test3
Source Acc (%)	...		
Target Acc (%)	...		

dann

	Test1	Test2	Test3
Source Acc (%)	...		
Target Acc (%)	...		

- Q2 (5%): Write your own codes to **visualize** the feature space by using the TSNE(perplexity=30, n_components=2, init='pca', n_iter=3000). Plot the feature distributions for both (1) original MNIST and MNIST-M inputs and (2) after DABP using source only. (3) after DABP using dann. (You will find useful functions inside the `utils`function. The example plot is shown below.)



input_tsne_plots

- Q3 (10%): Discussions
  (1) From the results in Q2, are the both domains **closer/farther after performing the transformation**? If the answer is closer, it verifies that DABP can learn discriminative and domain invariant features. If not, explain your reasons.
  (2) List **one** of the main problems for the DABP method and **explain** why? (e.g., the gradient reversal layer, weights shared for both source domain and the target domain)

Problem 3: GAN (25%)

In this assignment, we will implement a generative adversarial network (GAN) to generate new celebrities after showing pictures of many real celebrities using the CelebA dataset.

Fig. 4: Some sample images in the CelebA face dataset.

(a) Prepare CelebA Dataset

1) Download the CelebA dataset from the link (~1GB) and upload to your Google Drive. Then, unzip the data to your Google Colab server.
2) Load the dataset into PyTorch dataloader and visualize some sample images in this dataset.

(b) GAN Implementation

The discriminator is made up of strided convolution layers. The input is a 3x64x64 input image and the output is a scalar probability that the input is from the real data distribution. The generator is composed of convolutional-transpose layers. The input is a latent vector *z*, that is drawn from a standard normal distribution $N(0, 1)$ and the output is a 3x64x64 RGB image. The strided conv-transpose layers allow the latent vector to be transformed into a volume with the same shape as an image.

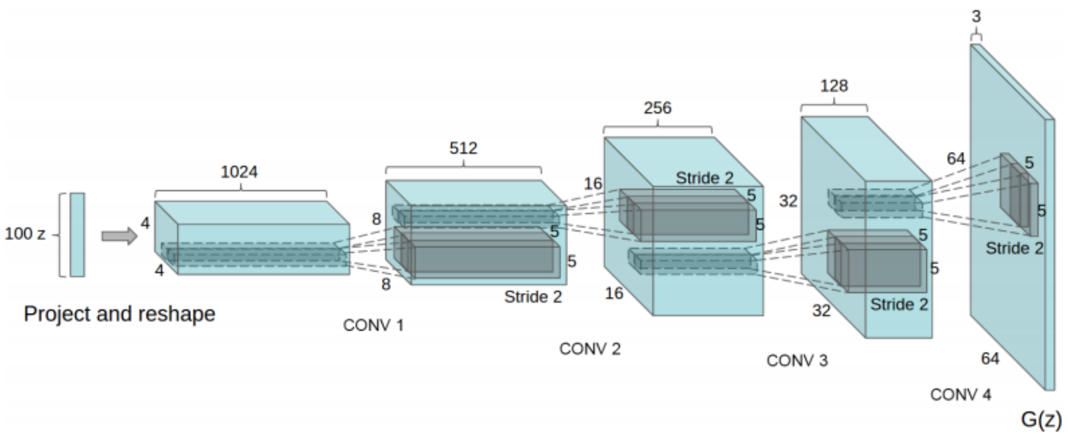


Fig. 5: The architecture for the GAN generator.

The details of the implementation are introduced in the notebook. Please read it carefully before working on this problem. The implementation of the GAN includes the following components (please follow the provided sample code):

- Weight Initialization
- Define Generator
- Define Discriminator
- Loss Functions and Optimizers
- Training Strategy

Please do the following:

1) Read the implementation code of GAN in the notebook.
2) Run the GAN and visualize the results using the code in the notebook.
3) Train for more epochs to see if the results can get better.
4) Change the size of (*output) images from 64x64 to 128x128, and retrain the network. Note that after changing the image size, some of the network parameters also need to be changed.

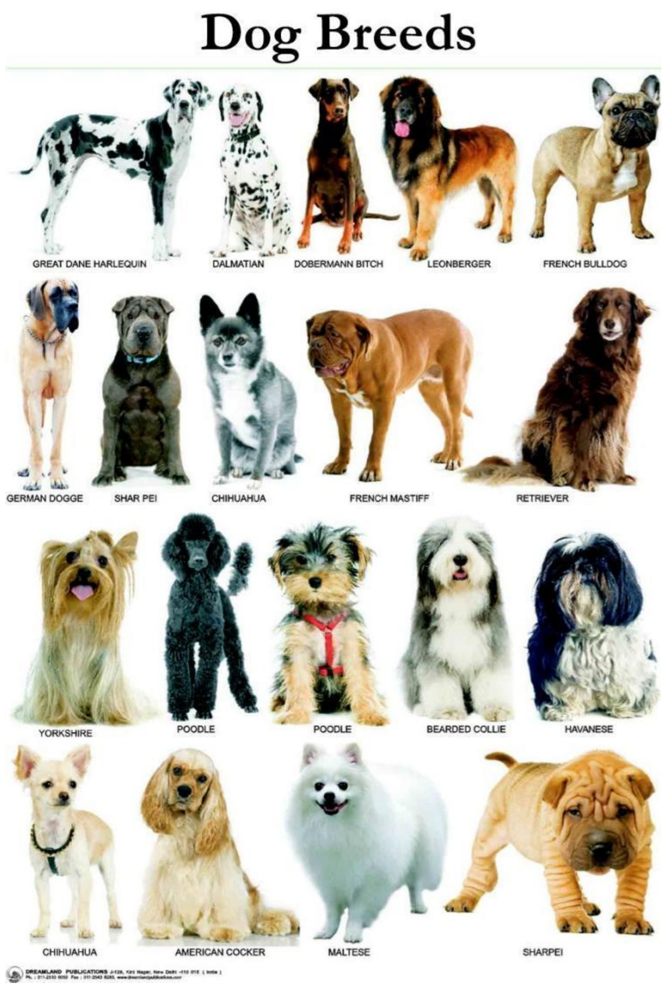
(c) Discussions

After GAN implementation, think about the answers for following questions:

1) Why does the training strategy for GAN need to be carefully designed?
2) Besides generating fake images from the noise, what other applications can you come up with for GAN?

Problem 4: Questions and Answers (25%)

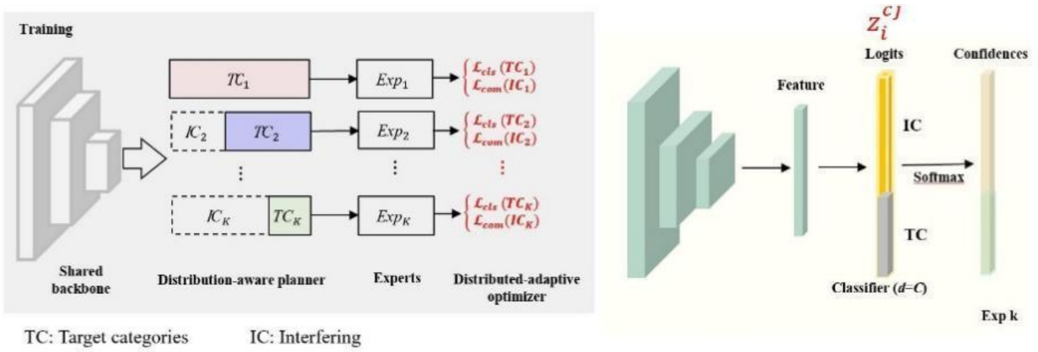
1. (5%) If I want to design a CNN based image classifier to recognize the following 19 different kinds of dog breeds (great dane harlequin, dalmatian, dobermann bitch, leonberger, French bulldog, german dodge, etc) , as well as their sex (male and female), 8 hair colors and 3 different sizes (small, medium and large), i.e., each image needs to find one class from the above four attributes (breed, sex, color and size), but not all the possible combinations are available (do not design a system with 19x2x8x3=912 output classes), how will you design such a system?



2. (5%) Face Verification: Given an AlexNet, pre-trained by 1000-class ImageNet dataset, and the Oxford VGG 2622-identity Celebrity Face Dataset, you want to use this AlexNet to train a face verification system to design a camera-based automatic gate entry system for the 20 students in your lab, where you have collected one photo from each of them. Please describe the training and inference procedures, and the corresponding loss functions, of your designed system

3. (5%) In Ally Complimentary Experts (ACE) method (see Figure below), what is the purpose of introducing the regularization term as shown in the overall loss function. Why not just simply force all the IC outputs to be zero since we are only interested in TC outputs? Without this term, what will be the bad side effect?

$$\sum_{c_j}^{\{IC_i\}} \|z_i^{cj}\|^2$$

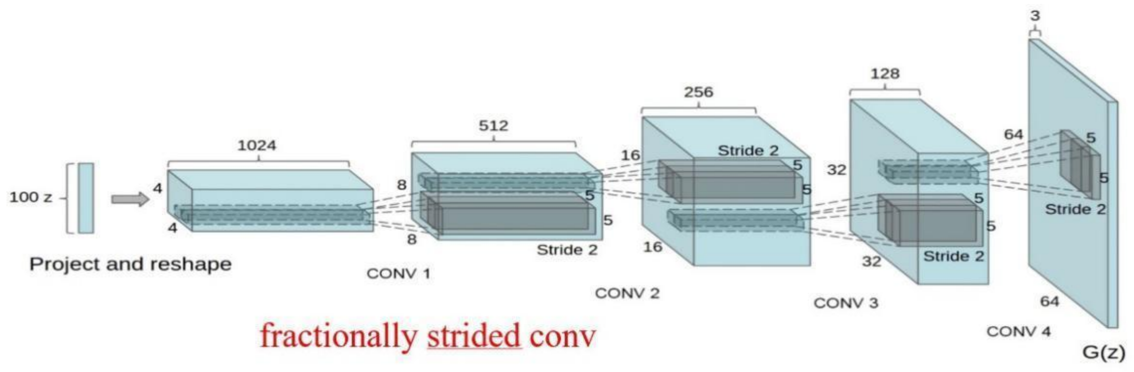


Training
TC_1
IC_2
TC_2
IC_K
TC_K
Exp_1
Exp_2
Exp_K
$\mathcal{L}_{cls}(TC_1)$ $\mathcal{L}_{com}(IC_1)$
$\mathcal{L}_{cls}(TC_2)$ $\mathcal{L}_{com}(IC_2)$
$\mathcal{L}_{cls}(TC_K)$ $\mathcal{L}_{com}(IC_K)$
Shared backbone
Distribution-aware planner
Experts
Distributed-adaptive optimizer
z_i^{cj}
Logits
Confidences
Feature
IC
Softmax
TC
Classifier (d–C)
Exp k

TC: Target categories IC: Interfering categories

$$L^i = L^i_{cls} + L^i_{com} = \overbrace{-\sum^{\{TC_i\}} y\log(\sigma(z_i))}^{\text{Cross-entropy loss}} + \overbrace{\sum_{c_j}^{\{IC_i\}} \left\|z_i^{cj}\right\|^2}^{\text{Regularization}}$$

4. (5%) The generator network used in the anime face generation, as shown below, takes a vector of 100 random numbers drawn from a uniform distribution as input and outputs an anime face image of size 64 × 64 × 3. The network architecture consists of a fully connected layer reshaped to size 4 × 4 × 1024 and four fractionally-strided convolutional layers to up-sample the image with a 5 × 5 kernel size. A fractionally-strided convolution (known also as 'deconvolution') can be interpreted as expanding the pixels by inserting zeros in between them. Please determine how many trainable parameters are involved in this architecture?



5. (5%) During the autonomous driving, we have to face various bad weather condition, such as foggy days or raining days. Can you design a de-fogging scheme to systematically remove fog systematically from foggy image frames? What kind of data will you collect and what kind of training procedures will you design?