**Homework 2 (100%) due: TBD**

**Submission Instructions**

You will need to submit the following materials:

1. A zip folder containing all ipynb files with your results for all the problems. Answer the discussion questions in the notebook files. 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. HW2_George_Clooney.zip**,** including all ipynb files with answers (code, log, results and discussions).

**Problem 1: Image Classification by CNN (30%)**

In this problem, we will solve a simple image classification problem on the CIFAR-10 dataset. It has the classes: 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'. The images in CIFAR-10 are of size 3x32x32, i.e. 3-channel color images of 32x32 pixels in size. There are 50000 images in the training set and 10000 images in the testing set.

Please follow the instructions in problem1.ipynb to finish this problem.

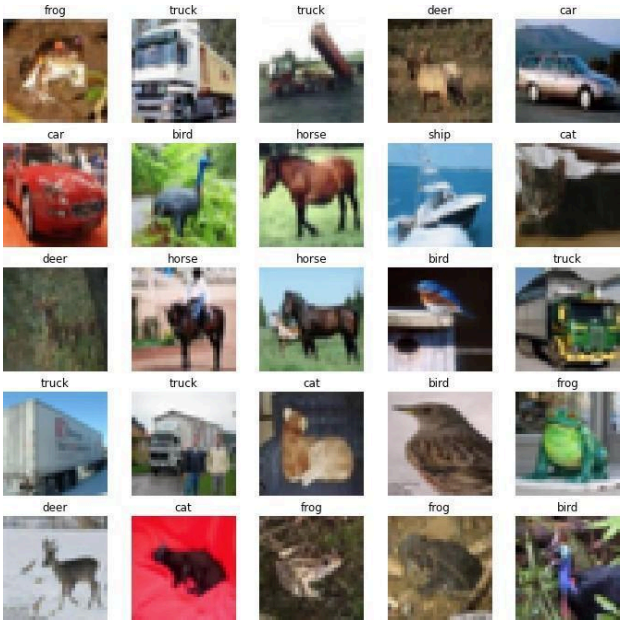



Fig. 1: Some sample images in CIFAR-10 dataset.

**Problem 2: Long-Tailed Recognition (40%)**

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. This imbalanced data distribution in the training set may largely degrade the performance of the machine learning or deep learning-based method.

Our goal is to build a CNN model that can accurately classify the images into their respective categories under imbalanced settings.

Dataset: Imbalanced CIFAR-30 ($\beta = 100$)

The CIFAR-100 dataset (Canadian Institute for Advanced Research, 100 classes) is a subset of the Tiny Images dataset and consists of 60000 32x32 color images. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. There are 600 images per class. There are 500 training images and 100 testing images per class.

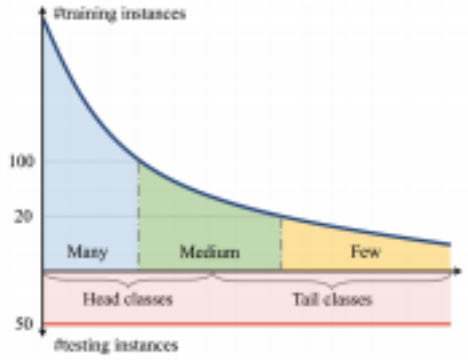



Figure 3: Our task of long-tailed recognition must learn from long-tail distributed training data and deal with imbalanced classification over the entire spectrum.

You will be building the imbalanced version of CIFAR-30 using the code we provide and the imbalance factor $\beta$ is defined as:

$$\beta = \frac{max(\{n1, n2, \cdots, nk\})}{min(\{n1, n2, \cdots, nk\})}$$

where $n_i$ represents the number of images for class $i$. Therefore, the larger the imbalance factor $\beta$ is, the harder it gets for doing long-tailed recognition on such data. With a $\beta = 100$ version of CIFAR-100, the head classes will have 500 training samples while the tail classes only have 5 training samples.

2-(a). [10%] Build and train a CNN for a balanced CIFAR-30 dataset and an imbalanced CIFAR-30 dataset, and report the accuracies and your observations in the notebook.

2-(b). [15%] Implement re-sampling for imbalanced CIFAR-30 dataset, and report the accuracies and your observations in the notebook.

2-(c). [15%] Implement re-weighting for imbalanced CIFAR-30 dataset, and report the accuracies and your observations in the notebook.

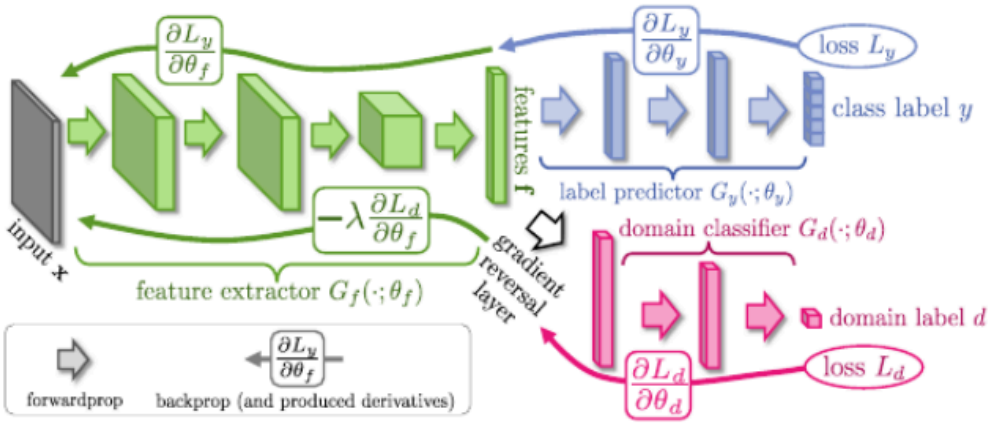
NOTE THAT THE CREDIT WILL BE GIVEN BASED ON THE CORRECTNESS OF YOUR IMPLEMENTATION! INCORRECT IMPLEMENTATION WILL NOT GET ANY CREDIT!

**Problem 3: Transfer Learning and Domain Adaptation (30%)**

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).

Problem Statement

Given a labeled 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 labeled. 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 labeled.
● 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.

Q0:

Step1. Follow the instructions in the notebook to download the pre-processed MNIST and MNIST-M dataset and visualize some examples.

Step2. 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

Step3. Train the DABP model by running `main.py` and answer the following questions.

Q1:

Perform 3 experiments on training and report your source and target accuracy in the tables in the notebook. (Your result is the average of the Target Accs. based on 3 experiments)

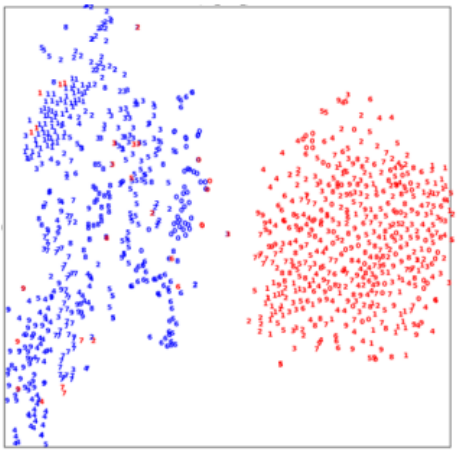
source_only

| | 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

*There is still Q3 on the next page!*

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)