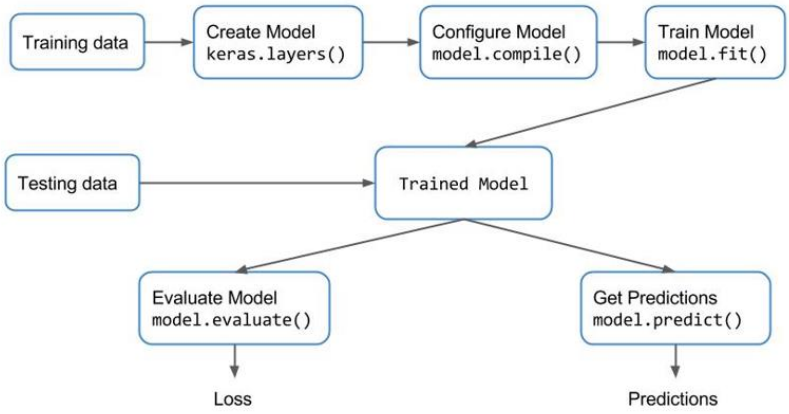
CSC 84020 Neural Networks and Deep Learning Spring'21

HW2 Classification and Prediction with DL Multilayered Perceptron

1. Requirements to the dataset. It should be a benchmark dataset (not generated one) for classification. The **number of classes should be not less than 5 and** the total number of patterns (training + validation + testing) **should be not less than 24,000**. Each team must use different dataset and there should not be repetitions. Please, send me its name, the details and web link of the dataset chosen by you for preapproval. The principle will be first come, first serve. After preapproval, you need to get different statistics of the dataset (HW1) and analyze the results.
Provide Report1 (pdf or doc file). **(10 points)**
(**MNIST as well as MNIST like are excluded as a choice for HW2**).

2. DL MLP topology minimum requirements. The task is to train a machine learning algorithm to recognize a new sample from the test set correctly using feedforward Deep Learning Multilayered Perceptron (DLMLP). The block diagram is given here only for reference. The number of hidden layers should be not less than 5. Normalize the dataset you use. You choose the activation functions and the number of neurons in each layer. For weight initialization I suggest Glorot Normal.

NOTE: For loss function use categorical crossentropy (not binary crossentropy) and for the last layer - softmax as it is a multiclass classification problem. Set the #epochs to 30 – 40. Set the batch size to 32 or 64 or 128 (the last figure if the total number of your dataset patterns is close or above 60000).



3. Training, prediction. Use Sequential model, mini-batches and mini-batch normalization, plot the loss and accuracy curves for the cases (3a – 3c) specified below. Add regularization and dropout to the model to prevent overfitting and check the performance again. Get the predicted class. Evaluate the test results.

(max 160 points)

NOTE: For each case below (3a – 3c) submit the code (3 files: py, inbpy and pdf for each case), the results and result analysis.

3a) Dataset Descriptive Statistics and Analysis. Analysis should be no shorter than one page, Times New Roman, font 11, single space. **(max 20 points)**

3b) Normalize the dataset before you start training. Split the dataset into train, validate and test (70 % - 20% - 10%). Some datasets offer these three subsets, so you can use them directly as they are. Include the table and graphics presentations shown in the template for descriptive statistics (part 12. Classification). Use SGD with Nesterov Momentum. Choose activation function.

Make experiments with dropout and dropout plus regularization (shown below) with red color OR without dropout and dropout plus regularization (shown below) with red color.

Evaluate test results using confusion matrix, classification report (precision, recall, f1-score, support, ROC/AUC and Precision-Recall curves). Except tables for this evaluation provide graphics of normalized confusion matrix, ROC/AUC curves. Add result analysis (not less than one page, font Times New Roman, size 11, single space.

(max 70 points)

3c) Use one of the optimizers (Adam, AdaMax, Nadam, AMSGrad etc). Choose activation function different than the one you use in 3b).

Make experiments with dropout and dropout plus regularization (shown below) with red color OR without dropout and dropout plus regularization (shown below) with red color.

Evaluate test results using normalized confusion matrix, the rest of the measures that you already have, ROC and AUC. Except tables for this evaluation provide graphics of normalized confusion matrix, ROC/AUC and Precision-Recall curves. Add result analysis (not less than one page, font Times New Roman, size 11, single space.

(max 70 points)

4. Discussions, Conclusions (not less than 2 pages without repeating any of the above two analysis, more stress on Comparison and Suggestions for improvement (single space, Times New Roman font 11). Do not forget References.

(max 30 points)

Total Max Number of points for completion of (1-4): 200 points

==========

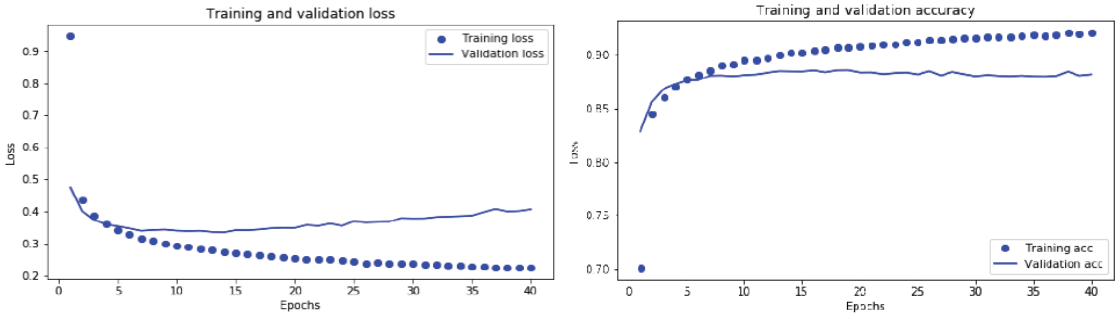
Details about dropout and L2 norm regularization.

```
"""## Dropout"""

def build_model():
    model = models.Sequential()
    model.add(layers.Dense(64, activation='relu', kernel_initializer='glorot_normal',
bias_initializer='zeros', input_shape=(128,)))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(32, activation='relu', kernel_initializer='glorot_normal',
bias_initializer='zeros'))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(32, activation='relu', kernel_initializer='glorot_normal',
bias_initializer='zeros'))
    model.add(layers.Dense(32, activation='relu', kernel_initializer='glorot_normal',
bias_initializer='zeros'))
    model.add(layers.Dropout(0.4))
    model.add(layers.Dense(10, activation='softmax'))
    optimizer = keras.optimizers.RMSprop(lr=0.0005, rho=0)
```

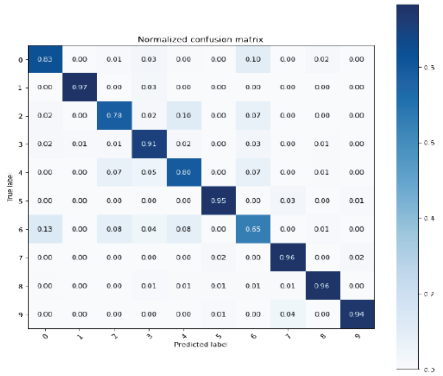
```
    model.compile(optimizer=optimizer,
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model


"""## Dropout + L2 norm"""
def build_model():
    model = models.Sequential()
    model.add(layers.Dense(64, activation='relu', kernel_initializer='glorot_normal',
bias_initializer='zeros', input_shape=(128,), kernel_regularizer=regularizers.l2(0.0005)))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(32, activation='relu', kernel_initializer='glorot_normal',
bias_initializer='zeros', kernel_regularizer=regularizers.l2(0.0005)))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(32, activation='relu', kernel_initializer='glorot_normal',
bias_initializer='zeros', kernel_regularizer=regularizers.l2(0.0005)))
    model.add(layers.Dense(32, activation='relu', kernel_initializer='glorot_normal',
bias_initializer='zeros', kernel_regularizer=regularizers.l2(0.0005)))
    model.add(layers.Dropout(0.4))
    model.add(layers.Dense(10, activation='softmax'))
    optimizer = keras.optimizers.RMSprop(lr=0.0005, rho=0)

    model.compile(optimizer=optimizer,
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

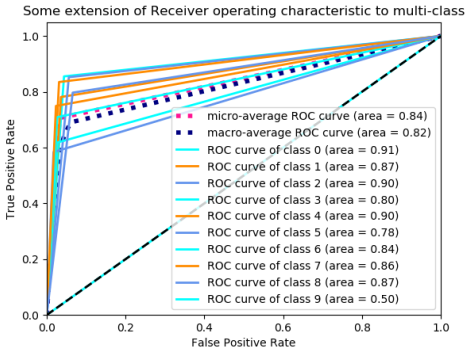
==========

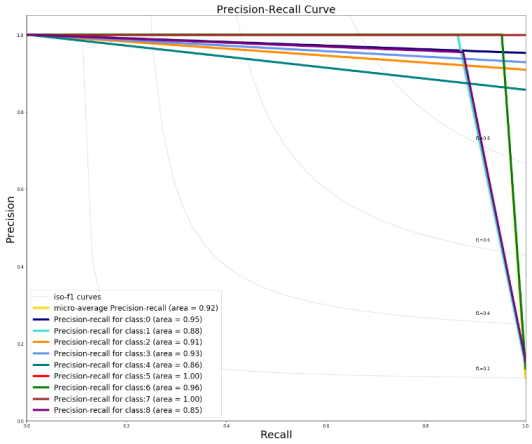
Details about the graphics: some examples are included:



Or you may use different colors and replace dots with a curve. There must be appropriate notation.







<u>Submission:</u>

a) **Report1 in doc or pdf format plus topology of your DL MLP: number of layers, number of neurons in each layer, activation functions and optimizers you use. Also the values of some NN parameters.**

b) **Upload your Python file1 (.zip) (Solution 3a) which combines your program code (.py, .ipynb and converted to pdf .ipynb), all your outputs and your analysis. Make sure to include comments.**

c) **Upload your Python file2 (.zip) (Solution 3b) which combines your program code (.py, .ipynb and converted to pdf .ipynb), all your outputs and your analysis. Make sure to include comments.**

d) **Upload your Python file3 (.zip) (Solution 3c) which combines your program code (.py, .ipynb and converted to pdf .ipynb), all your outputs and your analysis. Make sure to include comments.**

e) **Upload your part 4 (Conclusions, Discussions, References) in doc or pdf format.**

f) **List all members of your team as well as course number and Hw2 under comments. There is no need for each member of the team to upload the Hw2 on the BB. But make sure that the submission is done before the expiration of due date and time.**