# DL Assignment-Week 1
# Mar 06, 2024
# Submission Date: March 13, 2024 [14:00 hrs]

**==========Instructions==========**

5. You are not allowed to use any external libraries other than pytorch and required libraries for preprocessing, scikit-learn, scipy, numpy, pandas etc. **Violating this rule will attract up to a 100% penalty.**
6. **Submission Format:** A zip file with the name **DL-Week1_<RollNo>.zip** containing deliverables as discussed under the **Deliverables** section.
7. Plagiarism will lead to **100% penalty** to all the involved parties. You are allowed to discuss among each other but no code artifact is to be shared among each other.
8. Scoring will be leaderboard best. Try to make the best-effort submission. Leaderboard will be made on a held out dataset and will be made public only after the deadline has passed.

**==========**

This week, we are looking into Deep Learning models, an extension to previous week's assignment. The end goal is to build a classifier, which will label a social media post as either containing real and verified information or fake misinformation.

**Dataset.**
The dataset is from a shared task of Constraint@AAAI-2021. This task focuses on the detection of COVID19-related fake news in English. The sources of data are various social-media platforms such as Twitter, Facebook, Instagram, etc. Given a social media post, the objective of the shared task is to classify it into either fake or real news. It contains 10600 samples with 5545 labeled as real and 5055 labeled as fake.

`If you take Crocin thrice a day you are safe.` **Fake**

`Wearing mask can protect you from the virus` **Real**

We now describe the specific tasks you are expected to do in this assignment**.**

We will be working with 3 types of DL models:

- Deep Neural Network;
- CNN;
- LSTM.

**Task-1: Prepare dataset.**
Use scikit-learn train_test_split with shuffle option to split the dataset into training, validation, and test split. The fraction should be 80/10/10 (train/val/test).

**Task-2: Preprocessing Social Media Post.**
Social media posts contain a lot of crucial information which is not in text-format like emojis, urls, hashtags. You need to carefully preprocess such social media text because filtering out emojis or hashtags might flip the stance of the post. You must go through the following tutorial to understand how you can preprocess social media text.

**Task-3: Obtaining vector representations.**

For Deep Neural Network, we use tf-idf representation to encode the input sentences to vectors which are then fed into the model.

For CNN and LSTM, we build matrix representation of size: ***[max_inp_len,d]*** using a fasttext model trained on train+valid split; where max_inp_len is the size of maximum length input data point in the training dataset and d is the dimension of embeddings as output by fasttext model.

The process of constructing this matrix for each datapoint is as follows:

1. Initialize the matrix with the size as specified with zero.
2. Tokenize the input sentence
3. For each token with index **i** in the input, put the embedding of the token at index **i**.

**Task-4a: Training Deep Learning Classification models.**

You are supposed to build the following basic DL binary classification models, that takes a social media post and classifies into 0:fake and 1:real.

1. Deep Neural Networks
2. Convolutional Neural Networks
3. Long-Short Term Memory Networks

**Task-4b: Hyperparameter Tuning**

These DL models are very sensitive to the choice of hyperparameters that these DL models are initialized with. We describe three categories of hyperparameters.

1. Common hyperparameters for all the networks:
    a. learning rates,
    b. batch size,
    c. Learning rate schedule
    d. Optimizer
    e. Choice of activation function
    f. Whether to use EarlyStopping and with what patience
    g. #epochs
    h. Weight initialization
    i. Gradient accumulation steps
    j. Regularization

2. Extra Hyperparameters for DNN:
    a. #of layers,
    b. #of nodes/layers,
    c. dropout probability,

3. Extra Hyperparameters for CNN:
    a. Size of layers,
    b. size of kernel filter,
    c. stride step,
    d. padding,

e. BatchNorm vs LayerNorm,
f. Hyperparameters of DNN as described in point 2 for Feed Forward Dense Network.

4. For LSTM you can check out the following medium article.

You are expected to tune the hyperparameters involved in each of the Machine Learning Models. This might sound like a humongous task, and you might wanna check out Optuna for hyperparameter optimization task.

**Task-5: Evaluating Machine Learning Models.**

Now that your machine learning models are tuned and ready for evaluation, prepare a report that includes confusion matrix, classification accuracy, f1-score, precision, and recall (both micro and macro) on the test split that you obtained in Task-1. You can use the classification_report routine of scikit-learn for this task, but for sake of understanding we would recommend writing a routine of your own. Also include the final best set of hyperparameters obtained specific to each model in the report.

**Deliverables:**

1. 1 python file containing code for splitting, preprocessing, and vectorizing the input raw data. —**PreprocessAndVectorize.py**
2. 3 python files containing code for training each of the DL model. — **DNN.py, CNN.py, LSTM.py**
3. 1 python file containing code for running the evaluation which takes as input the path and type of DL model as cmd args. —**RunEval.py**
4. Saved model checkpoints on google drive, link to which should be set to 'Anyone with the link' with 'editor' privileges.
5. A shell script that sequentially runs all the task and generates the classifcation_report output at the end. —**runEndtoEnd.sh**
6. A report describing the hyperparameters and output of classification_report and google drive link containing saved model checkpoints.
7. Additionally write a python file that does following [**EvalTestCustom.py**] on the held-out test set:
    a. takes as input a csv containing social media posts and its corresponding label, the DL model type, the path to saved DL model from cmd.
    b. does the preprocessing and vectorizes it based on the input model type (**no splitting should happen!**)
    c. Loads the DL model saved on the path passed through the cmd
    d. Runs the inference on the vectorized input through the DL model
    e. Reports the classification_report output on the cmd considering predictions and gold labels.