1. **Generative Models for Text**

   (a) In this problem, we are trying to build a generative model to mimic the writing style of prominent British Mathematician, Philosopher, prolific writer, and political activist, Bertrand Russell.

   (b) Download the following books from Project Gutenberg `http://www.gutenberg.org/ebooks/author/355` in text format:

      i. The Problems of Philosophy
      ii. The Analysis of Mind
      iii. Mysticism and Logic and Other Essays
      iv. Our Knowledge of the External World as a Field for Scientific Method in Philosophy

   Project Gutenberg adds a standard header and footer to each book and this is not part of the original text. Open the file in a text editor and delete the header and footer.

   The header is obvious and ends with the text:
   *** START OF THIS PROJECT GUTENBERG EBOOK AN INQUIRY INTO MEANING AND TRUTH ***

   The footer is all of the text after the line of text that says:
   THE END

   To have a better model, it is strongly recommended that you download the following books from The Library of Congress `https://archive.org` and convert them to text files:

      i. The History of Western Philosophy
      `https://archive.org/details/westernphilosophy4`
      ii. The Analysis of Matter
      `https://archive.org/details/in.ernet.dli.2015.221533`
      iii. An Inquiry into Meaning and Truth
      `https://archive.org/details/BertrandRussell-AnInquaryIntoMeaningAndTruth`

   Try to only use the text of the books and throw away unwanted text before and after the text, although in a large corpus, these are considered as noise and should not make big problems.[1]

   (c) **LSTM:** Train an LSTM to mimic Russell's style and thoughts:

      i. Concatenate your text files to create a corpus of Russell's writings.
      ii. Use a character-level representation for this model by using extended ASCII that has $N = 256$ characters. Each character will be encoded into a an integer using its ASCII code. Rescale the integers to the range $[0, 1]$, because LSTM

[1] If this is a large corpus for your computer's power and it makes training LSTM hard, use as many of the books as possible.

uses a sigmoid activation function. LSTM will receive the rescaled integers as its input.[2]

iii. Choose a window size, e.g., $W = 100$.

iv. Inputs to the network will be the first $W-1 = 99$ characters of each sequence, and the output of the network will be the L^{th} character of the sequence. Basically, we are training the network to predict the each character using the 99 characters that precede it. Slide the window in strides of $S = 1$ on the text. For example, if $W = 5$ and $S = 1$ and we want to train the network with the sequence ABRACADABRA, The first input to the network will be ABRA and the corresponding output will be C. The second input will be BRAC and the second output will be A, etc.

v. Note that the output has to be encoded using a one-hot encoding scheme with $N = 256$ (or less) elements. This means that the network reads integers, but outputs a vector of $N = 256$ (or less) elements.

vi. Use a single hidden layer for the LSTM with $N = 256$ (or less) memory units.

vii. Use a Softmax output layer to yield a probability prediction for each of the characters between 0 and 1. This is actually a character classification problem with N classes. Choose log loss (cross entropy) as the objective function for the network (research what it menas).[3]

viii. We do not use a test dataset. We are using the whole training dataset to learn the probability of each character in a sequence. We are not seeking for a very accurate model of. Instead we are interested in a generalization of the dataset that can mimic the gist of the text.

ix. Choose a reasonable number of epochs for training (e.g., 30, although the network will need more epochs to yield a better model).

x. Use model checkpointing to keep the network weights to determine each time an improvement in loss is observed at the end of the epoch. Find the best set of weights in terms of loss.

xi. Use the network with the best weights to generate 1000 characters, using the following text as initialization of the network:

> There are those who take mental phenomena naively, just as they would physical phenomena. This school of psychologists tends not to emphasize the object.

xii. Extra Practice: Use one-hot encoding for the input sequence. Use a large number of epochs, e.g., 150. Add dropout to the network, and use a deeper LSTM (e.g. with 3 or more layers). Generate 3000 characters using the above initialization and report if you get more meaningful text.

xiii. Extra Practice- **HMM**: Train a Hidden Markov Model with V hidden states and V possible outputs using Baum-Welch Algorithm (or any other modern

[2]A smarter way is to parse the whole corpus to figure out how many distinct characters you have in the corpus (the number may be less than 256, e.g., 53). One can also disregard lowercase and uppercase letters or even remove punctuation characters such as !.

[3]In Keras, you can use the ADAM optimization algorithm for speed.

algorithm that is available) using the Russell corpus, where V is the number of distinct words in the corpus. Note that for HMM, you NOT use character level encoding, because it may yield totally meaningless results, although the transition matrices associated with it will be way smaller (you are welcome to try it). Generate 200 words using the model and comment on its meaningfulness. Extra extra practice: can you train a higher order HMM (i.e. an HMM that assumes dependency on more than one previous state) to get a better model?

2. **(Deep) CNNs for Image Colorization**

   (a) This assignment uses a convolutional neural network for image colorization which turns a grayscale image to a colored image.[4] By converting an image to grayscale, we loose color information, so converting a grayscale image back to a colored version is not an easy job. We will use the CIFAR-10 dataset. Downolad the dataset from `http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz`.

   (b) From the train and test dataset, extract the class birds. We will focus on this class, which has 6000 members.

   (c) Those 6000 images have $6000 \times 32 \times 32$ pixels. Choose at least 10% of the pixels randomly. It is strongly recommended that you choose a large number or all of the pixels. You will have between $P = 614400$ and $P = 6144000$ pixels. Each pixel is an RGB vector with three elements.

   (d) Run k-means clustering on the P vectors using $k = 4$. The centers of the clusters will be your main colors. Convert the colored images to k-color images by converting each pixel's value to the closest main color in terms of Euclidean distance. These are the outputs of your network, whose each pixel falls in one of those k classes.

   (e) Use any tool (e.g., openCV or scikit-learn) to obtain grayscale $32 \times 32 \times 1$ images from the original $32 \times 32 \times 3$ images. The grayscale images are inputs of your network.

   (f) Set up a deep convolutional neural network with two convolution layers and two MLP layers with. Use 5×5 filters and a softmax layer. Determine the number of filters, strides, and whether or not to use padding yourself. Use a minimum of one max pooling layer. Your input is a grayscale version of an image ($32 \times 32 \times 1$) and the output is a $32 \times 32 \times 3$ image that is colored using the main colors. Use a classification scheme, which means your output must determine one of the k color classes for each pixel in your grayscale image. Train at least for 5 epochs. Plot training, (validation), and test errors in each epoch. Report the train and test errors and visually compare the artificially colored versions of the first 10 images in the test set with the original images.

[4] MATLAB seems to have an easy to use CNN library. `https://www.mathworks.com/help/nnet/examples/train-a-convolutional-neural-network-for-regression.html`

(g) Extra Practice: Repeat the whole exercise with $k = 16, 24, 32$ colors if your computer can handle the computations.