OSE Questions

Name: Sneha Roy
Matriculation Number: 50078578

1.1 List five different tasks that belong to the field of natural language processing.

Five different tasks that belong to the field of natural language processing are text classification, machine translation, Named Entity Recognition(NER)(which includes the task of identifying and classifying entities such as organizations, people etc.) , text generation(includes predicting the next word in the sentence, text summarization), question answering.

1.2 What is the fundamental difference between econometrics/statistics and supervised machine learning?

The main fundamental difference between econometrics/statistics with supervised machine learning is that in econometrics and statistics, our aim is to understand the relationship between the variables and the causality. We want to draw meaningful insights and patterns from the data, and we are also able to test hypotheses, parameter estimation and make predictions regarding a particular domain of economics or social science. On the other hand, supervised machine learning trains on labelled data, to try to draw meaningful patterns from that data and make predictions or classifications on new, unseen data. The focus is mainly on creating accurate, predictive models that generalize well on unseen data. The success of the different supervised machine learning models depends on how well they perform on new data, it usually does not concern with dealing with the relationship between the variables or causality inference.

1.3 Can you use stochastic gradient to tune the hyperparameters of a random forrest. If not, why?

NO, stochastic gradient cannot be used to tune the hyperparameters of a random forrest. In stochastic gradient, we adjust the weights and coefficients to minimise the loss function on the basis of the training data in order to reach the global minima. Stochastic gradient descent is mostly used for neural networks and linear model, where there is a well defined loss function, which we minimise to get to the global minima. But on the other hand, random forrest has many trees, the construction of which does not involve minimising a single loss function that stochastic gradient descent can take care of. The hyperparameters of the random forrest that contains the number of trees, maximum depth, and feature selection criteria, do not have the gradient based optimization structure. It uses techniques like grid search, Bayesian optimization techniques. The collection of the decision trees in the random forrest, does not have single loss function to optimize. Random forests typically use techniques like grid search or random search for hyperparameter tuning, which do not involve gradient-based optimization.

1.4 What is imbalanced data and why can it be a problem in machine learning?

Imbalanced data is the situation where one class has significantly many more instances in the training set than any other classes.This can lead to the model being biased towards the majority class and low performance on the minority classes, since the model does not have a lot of instances to learn the underlying patterns of the minority class, thus it might lead to poor generalization in the minority classes. In some of the cases where the minority classes are of more importance like the detection of fraud or disease, imbalanced data can lead to low recall of the minority classes, because might will fail to identify minority classes. Accuracy is also misleading in this respect, because a model which only predicts the majority class might still have high accuracy but it is not a good model at all.

1.5 Why are samples split into training and test data in machine learning?

In case of machine learning or deep learning, it is always good divide the data set into train and test , in order to evaluate the performance of the model on the test set after being trained. In brief test data set helps to see the model's performance on the unseen data.
Looking at the performance of which can help to tune the parameters of the model. Overall, the training set is used to understand the patterns and relationships in the data, and the test set is mainly reserved to see how the patterns and relationships that we got from the test set generalize on new unseen data and it also helps us to detect problems like overfitting and others.

1.6 Describe the pros and cons of word and character level tokenization.

The word level tokenization presreves the meaning of the sentence, and they help to create the context of the input sequence, thus word level tokenization helps in understanding the meaning of the words and the relationships between them.It helps in the tasks of task classification, machine translation and others. Word level tokenization results in smaller vocabulary size results in easier computation. But word level tokenization is not good with out of vocabulary words. The character level characterization is good with out of vocabulary words,and certain languages with complex character structures. but it can't understand the meaning of words and their relationships, thus it is not useful for making context. Tokenized sequences are generally much longer at the character level, which can lead to increased computational and memory requirement.

1.7 Why does fine-tuning usually give you a better performing model than feature extraction?

Fine tuning is considered superior to feature extraction because it allows a pre-trained model to adapt specifically to a target task. While both techniques leverage pre-trained knowledge, fine tuning takes it a step further by adjusting the model's different parameters and weights to adapt with the task at hand. This adaptation leads to improved accuracy score especially if the new task at hand is significantly different from the original pre-trained task. In contrast, feature

extraction will use the pre- trained layers as fixed feature extractors with limited freedom to adjust the hyperparameters according to the task at hand.

1.8 What are the advantages over feature extraction over fine-tuning?

Feature extraction has a few advantages over fine-tuning, particularly when we are dealing with small datasets. One of the primary benefits is the ability to expedite the training process. With feature extraction, we can only change the custom layers specific to the task at hand, while keeping the other pre-trained layers intact. This significantly reduces the computational time and resources for training because it leverages the knowledge encoded in the pre-trained layers.
Feature extraction is also a preferred choice when the task at hand is kind of similar to the pre-trained task, again because it can leverage the knowledge encoded in the pre-trained layers and use it for the task at hand.

1.9 Why are neural networks trained on GPUs or other specialized hardware?

Neural Networks involve multiple extensive calculations. Given the GPUs ability to train large data sets and the processing of the train sets faster. We often run the models in GPU to train the large model much faster. GPUs have high bandwidth and so they excel in parallel processing, which is crucial for training neural networks with many parameters efficiently. Usually the model architecture of neural networks are very intricate and complex and it involves numerous matrix multiplications and operations, which GPUs can handle efficiently

1.10 How can you write pytorch code that uses a GPU if it is available but also runs on a laptop that does not have a GPU.

The code that I have used in my project that uses a GPU if it is available but also runs on a laptop that does not have GPU is:

```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
num_labels = 68
model = AutoModelForSequenceClassification.from_pretrained(
    model_name, num_labels=68
).to(device )
```

1.11 How many trainable parameters would the neural network in this video have if we remove the second hidden layer but leave it otherwise unchanged.

We are left with 784 inputs in and 16 neurons in the 3rd layer after we have dropped the 2nd layer. So the total number of weights from the input layer to the third layer is 784 * 16. The total number of bias trainable parameters is 16 (each for each neuron of the layer). Now the 3rd layer has 16 neurons, and the final output layer has 10. So the total number of weights

parameters would be 16 * 10. The total number of bias parameters would be 10 (each for each output) . Thus the total number of trainable parameters are 784 * 16 + 16 + 16 * 10 + 10 = 12,730 parameters.

1.12 Why are nonlinearities used in neural networks? Name at least three different nonlinearities.

Non linearities also known as activation functions in neural network introduce non-linearity into the model which helps the model to understand the complex relationships and patterns in the data. If the non linearities were not used, the model would only behave as a linear model which would significantly hinder the model's ability to learn complex patterns. Basically it enhances the model's ability to capture more patterns in the data. Non-linear activation functions re also differentiable which is crucial for training the neural network during back-propagation to compute the gradients and update the weights during training.
Example include - ReLU, Sigmoid, Tanh, Softmax.

1.13 Some would say softmax is a bad name. What would be a better name and why?

The max in the softmax name might be misleading in the sense that the work of the method softmax is not to maximize the raw scores, it is just to normalize them and make them positive. There is no function of the max part from the softmax.
It basically takes the values of the logits and exponentiated it, to make it a positive figure. and then we get the probabilities of each class by divding that exp. of that logit value divided by the sum of all the exp of the logits. Hence I would just like to keep the name softpreds , where the soft part of the word does the normalizing work and since it is something to do with the predicted probabilities , I retained the word preds.

1.14 What is the purpose of Dataloaders in pytorch?

Dataloaders in PyTorch serve the purpose of efficiently loading and preprocessing datasets for training and evaluation. They achieve this by batching data, which is crucial for managing large datasets, shuffling data to improve model generalization, and allowing for parallel computation to speed up data loading. Dataloaders also assist in separating training and evaluation data, simplifying the process of feeding data into the model during different phases. Overall, they enhance training efficiency, making it possible to train deep learning models effectively while optimizing resource utilization.

1.15 Name a few different optimizers that are used to train deep neural network.

A few different optimiziers that are used to train deep neural networks are: AdamW, Adam, Stochastic Gradient Descent(SGD), RMSprop.

1.16 What happens when the batch size during the optimization is set too small?

It will always take the same time to run one epoch, as one epoch = batch size * number of batches. The batch size and number of batches are inversely related to each other keeping the training time for each epoch will remain same keeping every other thing constant.
In case of a large dataset, a smaller batch size can lead to slower convergence. The model has to update the weights and biases more frequently, small batch size are more sensitive to noise, so there can be possible overfitting, increased variability and hence poor generalization and also inefficient GPU utilization. It can also interfere with the degree of parallelism during training which is usually designed to work of relatively larger batches.
In case of a small dataset, a relatively smaller batch size is advantageous promoting efficient convergence and reduced variability.
In case of imbalanced datasets, a smaller batch size can give more importance to minority classes which can be overlooked in large datasets.

1.17 What happens when the batch size during optimization is set too large?

As I pointed out earlier the training time required to run one epoch will be the same, so batch size will not affect training time. If the batch size is set too large relative to the dataset, there can be some problems. The model will might converge to a flatter and wider minima in the loss landscape, which may result in poorer performance on the unseen data, leading to poor generalization. It might also prevent the model from exploring the loss landscape very efficiently. It might also lead to delayed convergence because sometimes the noise introduced by the relatively smaller sizes helps to escape the local minima and lead to efficient convergence. large batch sizes may require more memory, limiting the model's training on GPUs with limited memory.

1.18 Why can the feed forward neural network we implemented for image classification not be used for language modelling?

Image data consists of fixed-size grids of pixels, and each pixel's value is treated as an independent feature. This grid structure enables to train the neural networks as they can be flattened into a single vector and the relationship between pixels are not important. But on the other hand, language data is sequential, with words or characters following a specific order to convey meaning. The relationships between these elements depend heavily on their positions within the sequence, and ignoring this sequential structure will lead to a loss of meaning and the models will not be able to understand or create the context based on the given text which is crucial for most NLP tasks.

1.19 Why is encoder-decoder architecture used for machine translation (instead of the simpler encoder only architecture we used for the language modelling)

In case of an encoder model, like BERT and others, the model takes in the encoded text and creates a context based on the given text. That context is then used for the tasks like text and

intent classification and others. But this is where the functionality of an encoder model stop. The encoder-decoder model takes it a step further, by taking in the context generated by the encoder model and generate an output sentence from the retained context. In essence, encoder models understand the sequence of text and generate contextual representations. Encoder-decoder models, used in machine translation and other sequence-to-sequence tasks, rely on both an encoder and a decoder to understand the input sequence, retain context, and generate the output sequence. So an encoder only architecture is not enough for tasks like machine translation.

1.20 Is it a good idea to base your final project on a paper or blogpost from 2015? Why or why not?

It would depend on certain factors such relevance or advancement of the field since then. If there has been very less literature on the topic after 2015, it sometimes might be fine to base my final project on that. But as in most machine learning literature, we have seen that, a lot of findings or literature are founded every year, so it might not be the best option always to base my final project on a blog post from 2015 because, I would miss out on a lot of recent and updated methods or literature. We have seen that, the BERT model was originally founded in 2018, but since then there has been so much advancement and new and more sophisticated models have come on such ROBERTA, Distilbert. So if I base my final project on a blogpost from 2015, I might miss out on many capabilities of the new sophisticated models It is .mportant of staying up-to-date with the latest advancements to ensure your project benefits from state-of-the-art techniques.

1.21 Do you agree with the following sentence: To get the best model performance, you should train a model from scratch in Pytorch so you can influence every step of the process.

Training a model from scratch provides full control over customization, data preprocessing, and hyperparameters, enabling tailored solutions to specific tasks. However, it can be computationally expensive and time-consuming, demanding substantial data and resources.
In contrast, transfer learning with pre-trained models like those from PyTorch or Hugging Face offers advantages when data is limited. Leveraging models trained on extensive datasets can yield good results with less data and training time. It’s crucial to apply transfer learning with caution, as domain knowledge is required to adapt pre-trained models effectively. The choice depends on project requirements and constraints, emphasizing the importance of informed decision-making.

1.22 What is an example of an encoder-only model?

An example of an encoder-only-model is "BERT"(Bidirectional Encoder Representations from Transformers) model.

1.23 What is the vanishing gradient problem and how does it affect training?

The vanishing gradient problem arises when we want to train a deep neural network particularly with many layers. This happens when the gradients which are the derivatives of the loss function with respect to the model's parameters become close to 0, when they are being back-propagated through the layers during the back-propagation process. This can arise when we are dealing with activation functions such as tanh and sigmoid. This happens when these activation functions encounter a high input value, the corresponding slope to which becomes close to 0. Thus the gradient descent problem comes to a halt and the model cannot update the parameters effectively. This can lead to slow or stalled learning and in some bad cases, it might not converge at all.

1.24 Which model has a longer memory: RNN or Transformer?

The transformer models have longer memory. There are few reasons for this. Firstly, attention masks or self-attention mechanisms and positional embeddings capture the different positions of words or characters in a text in a sequential way which is not done by traditional RNNs. This specific character of the transformer helps to understand the context of a text in a much better way than RNN.

1.25 What is the fundamental component of the transformer architecture?

The fundamental component of the transformer architecture is the 'attention mechanism'. The attention mechanism helps us to capture the complex relationships between the token in an input sequence by assigning different weights or attention scores to each token. These weights determine how much attention is to be given to a particular token by other tokens. In this way, by attending to the most relevant tokens of the input sequence, the models can generate the context of the input sequence, enabling it to understand complex relationship between the tokens and dependencies for tasks like machine translation and text generation.