Deep Learning Questions - Y-Data DL Course Homework Assignment

Question 1

1. Cross entropy: Loss which measures the performance of a classification model whose output is a probability value between 0 and 1. It increases as the predicted probability diverges from the actual label and penalizes both types especially those predictions that are confident and wrong.
2. Gradient clipping: A common and relatively easy solution to the exploding gradients problem by changing the derivative of the error before propagating it backward through the network and using it to update the weights. It includes two approaches: rescaling the gradients given a chosen vector norm and clipping gradient values that exceed a preferred range.
3. Residual connections: Type of skip connection, which is a module that skip some layers in the neural network and feeds the output of one layer as the input to the next layers (instead of only the next one). They are used to allow gradients to flow through a network directly (forward and backward), without passing through non-linear activation functions which may cause them to explode or vanish.
4. Momentum: A variant of the stochastic gradient descent which replaces the gradient with a momentum (the exponential moving average of current and past gradients). It is known to speed up learning and to help not getting stuck in local minima.
5. Cyclic learning rate: Learning rate schedule which instead of monotonically decreasing the learning rate takes a lower bound and an upper bound and allows the learning rate to oscillate back and forth between these two bounds when training, slowly increasing and decreasing the learning rate after every batch update. It enables more freedom in the initial learning rate choices and breaking out of saddle points and local minima and leads to fewer learning rate tuning experiments along with near identical accuracy to hyperparameter tuning.
6. Dropout: Computationally cheap and effective regularization method to reduce overfitting and improve generalization error in deep neural networks, where during training, some number of layer outputs are randomly ignored or dropped out. It cause

each update to a layer during training to be performed with a different view of the configured layer.

7. The Inception module: The basic of idea is to perform convolution on an input, with different sizes of filters and pooling, concatenate the outputs and send to the next layer. One reason for that is to avoid overfitting by making the network wider rather than deeper.
8. Stride: A parameter of CNN's filter that modifies the amount of movement over the input. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time.
9. Bottleneck layer: A layer that contains few nodes compared to the previous layers. It can be used to obtain a representation of the input with reduced dimensionality, like the use of autoencoders with bottleneck layers for nonlinear dimensionality reduction.
10. 1x1 convolution: Uses a filter of 1×1 which produce an output with the same height and width as the input but only one channel. It's usages are: Dimensionality reduction/augmentation, reduce computational load by reducing parameter map, add additional non-linearity to the network, create deeper network through "Bottle-Neck" layer, create smaller CNN network which retains higher degree of accuracy.
11. DenseNet: NN of which each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers (Concatenation is used). Each layer is receiving a "collective knowledge" from all preceding layers (due to that the network can be thinner and compact, i.e., number of channels can be fewer).

Question 2

Pros and cons of using small and large batch sizes:

	small batch size	**large batch size**
pros	Noisy, so offering a regularizing effect and lower generalization error. Make it easier to fit one batch worth of training data in memory.	Faster progress in training.
cons	Each step may be less accurate so it may take longer to converge.	Requires more resources. Degradation in the quality of the model, as measured by its ability to generalize.

Question 3

Three 3×3 filters sequentially can replace a 7×7 filter. To apply one 7×7 kernel we will need $7 \cdot 7 = 49$ weights, while to apply three 3×3 kernels we will need $3 \cdot 3 \cdot 3 = 27$ weights.

Question 4

In RNN, for every time step of a sequence, we backpropagate from h[t] to h[t-1]. First the gradient will flow through the tanh gate and then to matrix multiplication gate. Whenever we backprop into matrix multiplication gate, the upstream gradient is multiplied by the transpose of the W matrix. This happens at every time step throughout the sequence.

When the sequence is very long the final expression for gradient on h[0] will involve many factors of this weight matrix. This will either lead to an exploding gradient problem or vanishing gradient problem.

For exploding gradients we can use gradient clipping so we remain with the diminishing gradients problem.

Question 5

Methods for augmenting textual data:

1. Word shuffle - the order of words in a sentence is changed to create a new sentence. It must be considered that the classification algorithms should be sensitive to the order of the words.
2. Word replacement - identifies the most similar word as an alternative for original word and replace it.
3. Transforming syntax tree - syntactic paraphrases of the source sentence are generated. Different kinds of syntactic transformations such as replacing active to passive voice, using noun instead of pronoun, adding adjectives and adverbs etc, can be used to create paraphrases.
4. Query expansion - finding new sentences similar to a given sentence by considering this latter sentence as query and searching for its answer. This can be done by using a search engine and defining original sentence as a query. Then the most relevant search results

are selected as new sentences, which would inherit the same tag as the original sentence (the query).

5. Spelling errors injection - generate texts containing common misspellings for the purpose of training the model so as to make them more robust for this type of textual noise.
6. Back translation - translate the text data to some language and then translate it back to the original language. This can help to generate textual data with different words while preserving the context of the text data.

Question 6

For explaining what CNN learn we can:

1. Plot the learned filters directly - the learned filters are simply weights, yet because of the specialized two-dimensional structure of the filters, the weight values have a spatial relationship to each other and plotting each filter as a two-dimensional image could be meaningful.
2. Visualize feature maps - The activation maps, called feature maps, capture the result of applying the filters to input, such as the input image or another feature map.
   The idea of visualizing a feature map for a specific input image would be to understand what features of the input are detected or preserved in the feature maps. The expectation would be that the feature maps close to the input detect small or fine-grained detail, whereas feature maps close to the output of the model capture more general features.
3. We can use tensorboard for visualizing activations, weights, data from training and more.

And of course we can analyze the predictions and probabilities the model outputs.

Question 7

The curse of dimensionality could appear due to high dimensional input data.

It seems like in neural networks, besides the original input dimensionality, we might encounter this problem with large number of parameters (like hidden layers which increase the number of nodes from the previous layer). However, since usually each of the neurons becomes sensitive to a particular relevant feature it actually performs a form of

dimensionality reduction, finding the relevant data for this feature. So even if we enlarge the number of parameters in the network we are more able to spot the specifications we need to solve our task.

Question 8

This architecture isn't fully connected, and neurons are connected to only some of the neurons in the layer below it. Based on the neurons links and aggregations in this architecture each word will be affected most by itself and the 2 words before and after it, and as words are farther from it in the sequence they will affect it less (but will still have some contribution).

Not giving all the words in the sequence the same weight when predicting the sentiment of one of them may decrease the affect of a word with high correlation to specific sentiment if it is not close enough or miss the affect of a word that it existence affect the sentence (like “not” in the beginning of the sentence) and so on. Also based on the diagram we are limited to 22 words that affect the prediction of the word sentiment.

On the other hand it has logic that closer words are more relevant for understanding each word and should have larger affect on it (like bigrams, or sometimes the beginning of the sentences is not entirely connected to the start of it) so it can bring to more accurate results. Also this architecture is lighter than fully connected one.

So to summarize, this is probably a solution that will work better on some tasks and worse on others based on the nature of the data and the task.

Question 9

While the network doesn’t learn at all we can:

1. First pre step – check that our custom dataset is reasonable an. Correct errors if possible, spot imbalances.
2. Train our model on a small amount of data without augmentations trying to overfit. If it doesn’t work we may have programming issues: Simplify the model as possible to ensure that the basics of the network works (weights being updated, data is loaded and processed correctly etc). Also check that the data is normalized correctly. We can then

gradually add the complexity and additions to the network and data and spot the point of which the network become problematic.

If we have a dataset that it is known to be solvable for our task we can try to run our network on this dataset.

3. If adding data causes problems, or the network works on other dataset we may have problems with the custom dataset. We need to make sure that our data is correctly normalized and preprocessed, shuffled during training, adjust batch size etc..
4. At this stage we hopefully spotted programming and major data issues so the network should learn something, even if it doesn't perform well. Now we can customize the data (like augmentations) and the network (like regularizations) to improve the results.

Question 10

The main problem with imbalanced dataset for deep learning is not getting optimized results for the class which is unbalanced in real time as the model never gets sufficient look at the underlying class.

Also, it creates a problem of making a validation or test sample as it is difficult to have representation across classes in case number of observation for few classes is extremely less

A few ways to avoid these problems (numbers 1, 2, 3 and 6 are relevant only to the first problem):

1. Weight balancing – instead of computing our loss function with equal weight to each training example we can enlarge the weight for examples from the minority class/es on the expense of the other classes. So they will have more effect on the loss function.
2. Focal loss – another intervention in the loss class weights. This time, instead of giving equal weighting to all training examples, we down-weights the well-classified examples. So where we have a data imbalance, our majority class will quickly become well-classified since we have much more data for it. And in order to ensure that we also achieve high accuracy on our minority class, we can use the focal loss to give those minority class examples more relative weight during training.
3. Undersampling - we will select only some of the data from the majority class, only using as many examples as the minority class has. This selection should be done to maintain the probability distribution of the class.

4. Oversampling - we will create copies of our minority class in order to have the same number of examples as the majority class has. The copies will be made such that the distribution of the minority class is maintained.
5. Synthetic data – if feasible, we will enlarge the minority class using slightly modified copies of the minority class/es data or synthetically new data created from the minority class/es data.
6. If it is a binary classification, we can adjust the decision threshold.

Question 11

Does the question means using the ImageNet as is without any fine tuning or not using ImageNet at all?

If the meaning is to use ImageNet as is:

1. I have a small dataset with images similar to ImageNet - all I need to do is to customize and modify the output layers according to my problem statement.
2. Attempting to finetune the model bring worse results.
3. I am still waiting for amazon to approve my GPU and Colab claims I exceeded the GPU limit so for now I don't have resources to any kind of model training.

If the meaning is not to use ImageNet at all:

1. My data set is very different from ImageNet and it better start with random weights.
2. ImageNet model architecture does not suit my task or data.
3. My images dimensions are very different from ImageNet input shape and cannot be adjusted without damaging the data.

Question 12

The idea will be to arrange the English and German embeddings to that similar words in both languages will be close to each other in the semantic space.

One approach can be to use multilingual BERT to produce embeddings out of the box for both the English and German emails. Then train the classifier with the English data and use it to predict the German data.

Another approach can be to train word2vec on both the English and German data, use a tool like transvec to provide translations and create a bilingual model based on the two model.

Then produce embeddings for the English and German data and again train on the English data and predict on the German.

If we prefer not to retrain the model and use it as it (maybe we don't have a labeled English data or the retrained model doesn't produce good enough results) we can find a translation function from the German embeddings to the English embedding space (again by training English word2vec and German word2vec and finding this function using some word translations). Then we can predict the German email using the "translated" German embeddings with our original model.

Question 13

Needed data: Hebrew sentences with full diacritization. The data needs to be as large as possible and come from different domain to enable generalization. We will clean the data and remove numbers and punctuation marks (even though it might change the context or a meaning of a word in a sentence). For each sentence we need to create a clean version of the sentence without diacritization.

Encoding: We set a code for each letter with or without a diacritization, so for example א, אִ, אָ will have different codes. Some letters can get 2 diacritizations so we need to set codes for those cases as well. Space will get a code of 0. We encode both the clean sentences which will serve as inputs and the sentences with diacritization which will serve as the output. We might have to set the sequences to a fixed length with cutting or padding each sentence if we wish to train with batches.

Another output approach: We can create a class for each diacritization mark or diacritization combination on one letter. Then the output is a vector of diacritizations classes where space or no diacritization has the "blank" class. We will need to create such vectors from the sentences with diacritization marks for training and validation.

Model: A proposed architecture may be:

- Create random embeddings for each char code that can change during training.
- BiLSTM layer.
- Linear layer with output size as num of classes (letters with diacritizations or just diacritizations based of the output method).

(Of course, the model can be changed and approved based on the training results, using dropouts, more layers, different LSTM, etc).

Training: Divide the data to train and evaluation. Apply softmax to get the letter diacritization probabilities. The model is trained to minimize the cross-entropy of this distribution with the target labels.

Evaluation: Analyze accuracy against the target sequence with:

- The proportion of characters with incorrectly restored diacritics.
- The percentage of incorrectly diacritized words (words which one letter in the word have a diacritization error).