Deep Learning Challenge

Intro:

Despite not being around for that long of a time, deep Learning models are now the standard tools in many fields, from Search Engines to Autonomous Vehicles. And now, time has come that the biology field is transformed, too !
At Proteinea, we use deep learning to power our protein design pipeline, helping generate and evaluate new proteins that could later be used as vaccines, therapeutics, enzymes ..etc
Our AI engineers have been focused on generative models for a long time now, and we would like to try out new ideas connecting deep learning and biology, and we need your technical skills to spearhead our efforts. We can't wait to see what you have got !

(70 points) Coding Problem:

For this problem, you are asked to go from a simple GAN (**Generative Adversarial Neural Network**) to a more complex GAN, implementing the Spectral Weight Normalization technique, in your deep learning framework of choice.

Technical Notes:

- For implementation, please use Python 3.6, choosing between Pytorch or TensorFlow 1 ***(do not use TensorFlow 2, avoid using keras)***. Please follow the hyperlinks for your corresponding choice.
- You will be using the MNIST digits dataset (Pytorch[**Link**], TensorFlow[**Link**])
- Start from those specific simple GAN implementations: (Pytorch[**Link**], TensorFlow[**Link**])
- You should start building your code from the simple code samples provided, modifying them as needed: either to organize your code, or to implement the task. Minimize using any code from stackoverflow, and definitely do not use ready-made code for the task.

Problem:

Starting from the simple GAN code provided for your framework (Pytorch or TensorFlow 1), please implement each of the following versions of the GAN incrementally, providing code and results (from middle and end of training) for each version:

1. **(5 points) A GAN with linear layers only. No weight normalization** (*Pytorch*: this means that you will have to actually remove the `BatchNorm1d` from your code and test it, *Tensorflow*: this means you will have to test the same code with the MNIST dataset. You may want to modify the architecture of your code to be like the Pytorch code's architecture.)

2. **(5 points) A GAN with linear layers and batch normalization**. (*Pytorch*: this means that you will have to run the sample code provided exactly as first given, *Tensorflow*: this means you will have to add 1D batch normalization to the code from the previous part)
3. **(20 points) A GAN with 2D CNN layers and 2D Batch Normalization.** Do not use flattened data, use the 2D version. For the architecture, you can get inspired from the DCGAN architecture here [Link], but you do not have to follow a certain architecture: you only have to make sure you have: 2D CNNs, 2D Batch Normalization, and improving results as training progresses. You are expected to use the batch normalization ready-made functions from your framework.
4. **(40 points) A GAN with 2D CNN layers and Spectral Normalization (instead of batch normalization):**
   Spectral normalization, similar to batch normalization, is a weight normalization technique, first suggested for use in deep learning in this paper [Link]. You may skim the paper for mathematical reasoning of why this works, but this is **NOT** relevant to the task. You are only expected to read *appendix A (page 15), specifically: Algorithm 1*, to get the gist of the algorithm. Implementation Examples are provided for (Pytorch[Link], TensorFlow[Link]).
   a. Don't get daunted by the paper's complexity, the algorithm itself is actually pretty simple. Reading the Implementation Examples will help you understand it.
   You are expected to replace the batch normalization from step 3 with spectral normalization, and report on your code and training results.
   b. Do **NOT** use the ready-made spectral normalization functions from your framework, and do not copy-paste the code from the code implementation examples. Write your own code, but it will naturally look similar to the provided examples since you are implementing the same algorithm.

Constraints (Restated, for clarity):

1. You should incrementally build on the simple GAN code provided (Pytorch[Link], TensorFlow[Link]). Please do not start from any other code online, this is to ensure a fair comparison between applicants.
2. You **can** use the ready-made batch normalization function in your framework
3. You can **NOT** use the ready-made spectral normalization function in your framework, nor copy online code implementing it. You should implement it yourself.

(20 points) Technical Questions:

(10 points) Question 1:

You have access to an infinite amount of GPUs that have 8 GBs of Memory each, with a deep learning model that you want to train. You try training your model on a batch of size 32, but to

your dismay, you get an OOM error (out of memory error) from your GPU. You keep decreasing the batch size, to find that you can only train on a batch size of 1 without going out of memory.

a) (2 points) How many GPUs will you need to train your model for a batch size of 32 ?

b) (4 points) What is the specific PyTorch or TensorFlow module you will use to conduct such a type of training? You may write a maximum of 280 characters about how you will use it.

c) (4 points) You connect the required GPUs, and try to train your model on them, but you face the exact OOM error again when you go beyond a batch size of 1. When you check the code, you realize the code has access only to one GPU. Given hardware is connected OK, what do you check next ? ***Hint: It's probably a software issue outside your deep learning library.*** Write a maximum of 280 characters elaborating on what you check next.

(10 points) Question 2:

A deep learning developer is working her way through the Simple MNIST Convnet Keras Tutorial [Link]. She opens the google colab from the tutorial, and runs the code. But to her surprise: she gets better accuracy than the tutorial from the first epoch! "Wonderful!", she thinks. On the next day, she tries the exact same code, but gets worse accuracy than the tutorial on the first epoch. She comes to you, the deep learning expert she knows, asking for an explanation for why this happened: why does the exact same code produce different results every time she runs it ? Write your explanation in no more than 400 characters.
Hint: This is not something specific to the Keras framework, it has to do with how deep learning libraries work in general.

Files to Submit:

1. Link to a public github Repo (preferred) or Zip file of your code. Please provide a readme of how to test/run your code. We should be able to replicate your experiments, and run each of the 4 versions of the code.
2. **Most preferably:** A conda spec file for Ubuntu 16.04, 18.04 or 20.04 on x86-64 architecture (**how-to link**)
   Other options (less preferred, but won't affect candidate's score):
   a. requirements.txt file generated by pip (**how-to link**)
   b. environment.yml file generated by conda package manager (**how-to link**).
3. Repor: PDF file named "Report.pdf", in which you answer the two technical questions. You don't have to add a section of the report for the coding problem, but in case you need an easy way to share figures or general comments, you may include them in a separate section in the pdf file.

Evaluation Criteria:

1. Points mentioned between brackets before each task or question denote its weight in the total task. More points = more weight = more important task or question.
2. For the code, you will not be evaluated for how good the training results look, but on the correctness of your implementation and how well you use the package you chose. Do not train for too long, do not do hyperparameter tuning nor change the loss function. You should just see results improving over training epochs and then turn off the training. Just make sure to train for the same number of epochs across all your trials to be able to compare the effect of different techniques **(Linear, Linear with batch normalization, CNN with batch normalization, CNN with spectral normalization).**
3. Code readability. This is necessary for us to be able to evaluate your code.
4. Results readability. Do not invest so much time in the display, but you must make sure to plot some samples from training steps in each of the four cases.

Inquiries:

1. For technical inquiries, contact: walid@proteinea.com
2. For logistic/time-limit inquiries, contact: salma.elalfy@proteinea.com