**Instructions**

Question 1.         [50 points]

In this problem, you will again use the Rock dataset located here: [https://osf.io/d6b9y/Links to an external site.](https://osf.io/d6b9y/).

**Extra tips:**

* If you are using Colab, make sure to select a GPU (even if you're not using a GPU via Edit->Notebook settings, it will give you more memory; no need for the paid version of Colab).
* You can convert images to grayscale from RGB and complete the assignment using only grayscale images.
* Feel free to downsize images if you are facing issues with the memory/running time.
* If you don't like Colab, you can also use Kaggle or similar tools ([https://www.kaggle.com/codeLinks to an external site.](https://www.kaggle.com/code))

Use transfer learning for image classification, going through these steps:

1. From the '360 rocks' folder create a training dataset that has 30 categories and 12 images in each category. To create the categories, group the rocks by name: e.g., one category will be 12 Andesite rocks, another category will be 12 Basalt rocks, etc.  From  '120 rocks' folder create a validation dataset that has 30 categories and 4 images in each category. **[10 points]**
2. Build the input pipeline, including the appropriate preprocessing operations, and add data augmentation. **[10 points]**
3. Fine-tune a pretrained neural network of your choice on the training data from the previous step. Before fine-tuning, you should remove the top layer of the pretrained network and add at least two more layers with a softmax activation function (your output layer should have 30 neurons and the layer before the output layer should have 8 neurons). For fine-tuning, you should first train for a few epochs only weights in the layers that you added **[5 points]**, and after that, you should train all the weights **[5 points]** (you should try to run your code until convergence, but for at least 20 epochs). Display train and validation loss and accuracy during the training and make a plot showing train and validation loss and accuracy as a function of the training epoch (mark the epoch where you switched to training the whole network with a vertical line) **[5 points]**. We will award the best performers with extra points. Discuss your findings - comment on the performance of the network **[1 point]**, convergence **[1 point]**, and possible hyperparameter tweaks you tried, or you would try to improve the performance **[3 points]**.
4. Compute the correlation coefficients between the network data and human data for each of the 8 neurons in your next-to-last layer (similar to HW3, using procrustes analysis) for the images from the train set (360 images) and (separately) for the images from the validation set (120 images). Report each of the 8 correlation coefficients and your average correlation coefficients (please mark in the bold with large font so we can easily find it). **[10 points]**

Question 2.         [50 points]

1. Create your own dataset for text classification. It should contain at least 2000 words in total and at least three categories with at least 100 examples per category (an example can be a poem or a paragraph from a book). You can create it by scraping the web or using some of the documents you have on your computer (do not use anything confidential) or ChatGPT. **[15 points]**
2. Split the dataset into training (at least 240examples) and test (at least 60 examples) sets. **[5 points]**
3. Fine-tune a pre-trained language model capable of generating text (e.g., GPT) that you can take, e.g., from the Hugging Face Transformers library with the dataset you created (this tutorial could be very helpful: https://huggingface.co/docs/transformers/training). **[20 points]**Report the testv accuracy **[5 points]**. Discuss what could be done to improve accuracy **[5 points]**.