Final Results

- Push your deliverable there as a pdf file: “Final Results.pdf”
- Push your codes there. If there’s more than one file, please create a new folder named “code”. Note: Make sure to maintain your/your team’s Github with properly documented README!

Deliverable Description In this deliverable, you will discuss your final training results and your integration approach. This is a machine learning course, so make sure to focus on the analysis of your results.

1. Final Training Results Compare your final results to your preliminary ones (from the previous deliverable). Have you changed anything to your model since the previous deliverable? If so, how have your changes improved the results? Now, focus on your final results. Once again, present a detailed analysis of your results, provide graphs as appropriate. Analysis requirements differ in every field, but must report at least one concrete metric relevant to the field in which you are working. Here are some examples: - Confusion matrix and accuracy/precision-recall/logistic loss (classification problems). - Mean squared error (regression problems) - Rand index (unsupervised models) - BLEU score with brevity penalty (text generation) - variance of the dimension reduced set vs variance of the initial dataset (dimensionality reduction/PCA)

   The proceeding results from the data fitting were proven to be better than the preliminary from the last deliverable. We were able to achieve 82% accuracy on the test set, compared to approximately 60% from the previous result.

   To achieve this new accuracy, we used a pretrained model from resnet50 (“imagenet” weights) which we fitted with our dataset. Also, we resized our images from 32x32 to 128x128, improving the results further. Additionally, we are working on evaluating the confusion matrix to be able to assess our model with an additional metric.

2. Final demonstration proposal: Now that you trained your model, it is time for you to integrate it in a final product. Don’t forget to save your trained weights!

You will need them for the integration and/or testing your model. (e.g. in keras: model.save_weights(filepath) and model.load_weights(filepath)) - Application We want all of you to at least have a landing page type website to demo your model and results. For more experienced developers, you are welcome to choose something more advanced. Discuss your final product, and final integration approach. Describe and justify the choice of stacks and technologies. Provide diagrams as appropriate. Explain your experiences with the technologies you have proposed. If you do not have any, explain how you would come to learn them (eg. online tutorials, etc.) If you do not have any experience creating a webapp and were not able to check out the MAIS workshop on integrating your ML model into a webapp, you can find the recording here: https://www.youtube.com/watch?v=aucqOA6kyiU&ab_channel=McGillArtificialIntelligenceSoc iety Here is the google drive with the resources mentioned in the video: https://drive.google.com/drive/folders/1MXVhrvm3QyH4WKLwG-B-C82yEPAJD62L?usp=sha ring And here is the GitHub repo for the basic webapp if you just want to start with the basic webapp and build yours from there: https://github.com/McGillAISociety/f2021-deploying-ml-mode

We are going to build a full-stack web app with Vue.js and Tailwind on the front end and then Flask on the back end.

We have experience in Vue and prefer it over React, and tailwind is also a CSS framework we have experience in that makes it very easy to style the front quickly and well so that it is responsive across platforms.

For the backend, we have less experience in Flask but it makes it easier to integrate with our python model since Flask is in python so we can just directly bring in our models and integrate with them through REST endpoints.

We plan to build a web app that provides a simple interface that lets the user take a photo and then we will send it to our backend, classify it, and send the classification to the user.

If we have time, we may build in account tracking or potentially a live, websocket based protocol, although the latter is unlikely.

Hosting will must like be done with Vercel for frontend and AWS for backend.