Selection and Optimization of Deep Convolutional Neural Networks as a Service

Erik Weissenborn[1*] & Yanis Tazi[1*]

[1] NYU, Computer Science - Deep learning systems

*equal contributions

Motivation:

Model selection, training, and finetuning of a Convolutional Neural Net is a time consuming and tedious task. It often requires multiple repetitions of training with various hyperparameter configurations. We decided to formalize this process by constructing a cloud service that would enable the user to upload a previously unseen labeled dataset and return to the user a highly accurate and finetuned CNN that is ready for use. The user will be emailed a report, which displays the fine tuned model's training and validation accuracy, training loss, and a confusion matrix with test results. The cloud solution offers an efficient way for users to obtain a fine tuned CNN model that they can begin using out of the box for inference.

Objective:

Build an efficient model selection and training process that can identify the optimal model and hyperparameters to use for a given dataset provided by a service user.

Architecture:

The model selection and optimization service will be built using Google Cloud services such as Google Storage, Google Functions, and Google Compute. The Google Storage service will be utilized to temporarily store user email and datasets. A storage bucket will also be used to store optimized models in Tensorflow file 'h5' format. These model files will be sent back to the user via email. The Google Function service will expose an HTTP endpoint that will be used as the endpoint for the service. The user will send their email as a parameter in the HTTP request and the function will store their email in Google Storage email bucket and return a signed URL that will be used to upload their dataset to the Google Storage dataset bucket.

The Google Compute service will have one running virtual machine that is using the Google Marketplace Deep Learning Image, which has Deep Learning frameworks Tensorflow and Keras, and Nvidia GPU drivers preloaded.

Workflow:

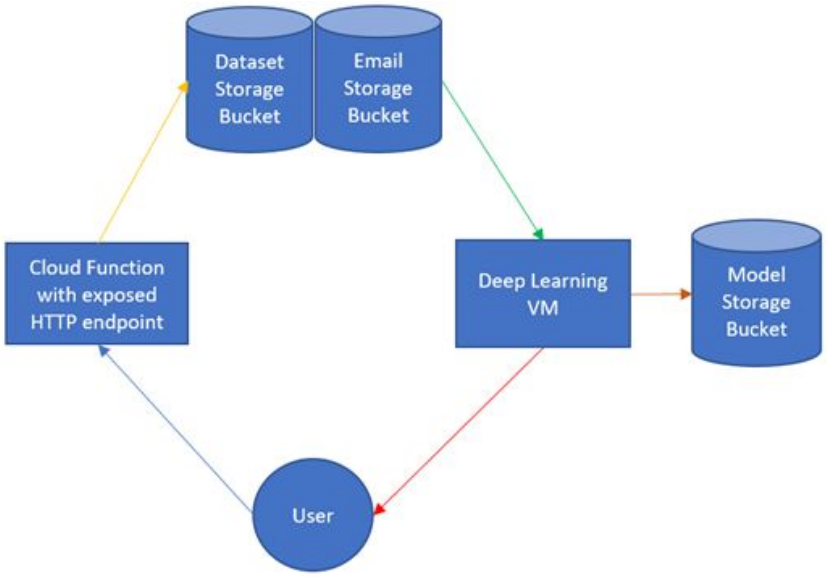
The virtual machine will poll the dataset storage bucket periodically to detect if any datasets have been uploaded. If a dataset has been uploaded, then the virtual machine will download the dataset zip file and user email from respective storage buckets and then process the dataset.

The dataset must be uploaded as a zip file with separate folders, each folder named as a classification label for the dataset. And within each folder will be the images used for training, validation and testing the DL models. For example, if a user would like a model to classify images of cats and dogs, they can upload a zip file containing ‘cat’ and ‘dog’ folders, each of which contain the population of respective ground truth images of cats or dogs.

During dataset processing, the virtual machine will run various high performing convolutional neural net models, which are pre-trained on Imagenet and are already available via the Keras API. Each of these models will have their base network layers frozen, while allowing their fully connected and prediction layers to train on the user’s dataset. Each model will train for a certain number of epochs, and the validation accuracy will be recorded. The model iterations with highest validation accuracy are saved in a local directory, and the best performing model is chosen for continued training and hyperparameter tuning. The best model is trained for another n number of epochs with a new set of hyperparameters to find a higher validation accuracy. Any new model iteration with a top validation accuracy is saved locally.

After training and hyperparameter tuning is complete, the test data is fed to the final model to produce the test accuracy. A confusion matrix and loss/accuracy graphs are produced and are emailed to the user along with the Google Storage URL of the final fine tuned model.

Workflow Diagram:



Neural Architecture Search:

There are numerous methods for selecting or creating the optimal neural architecture for a given dataset. A specific strategy must be chosen for exploring the architecture search space, which is vast. Each strategy researched requires a high number of training rounds, compute time, and combinations of different neural net components or preconstructed cells. Research has shown that these methods are sometimes no better than random selection of an architecture. Given that this task can consume considerable resources, we have chosen to use a subset of the highest performing, prevalent architectures, such as VGG16, MobileNet, Inception, Inception-Resnet, and Xception. We simply train each of these networks with the user's dataset and then choose the network that obtains the highest validation accuracy.

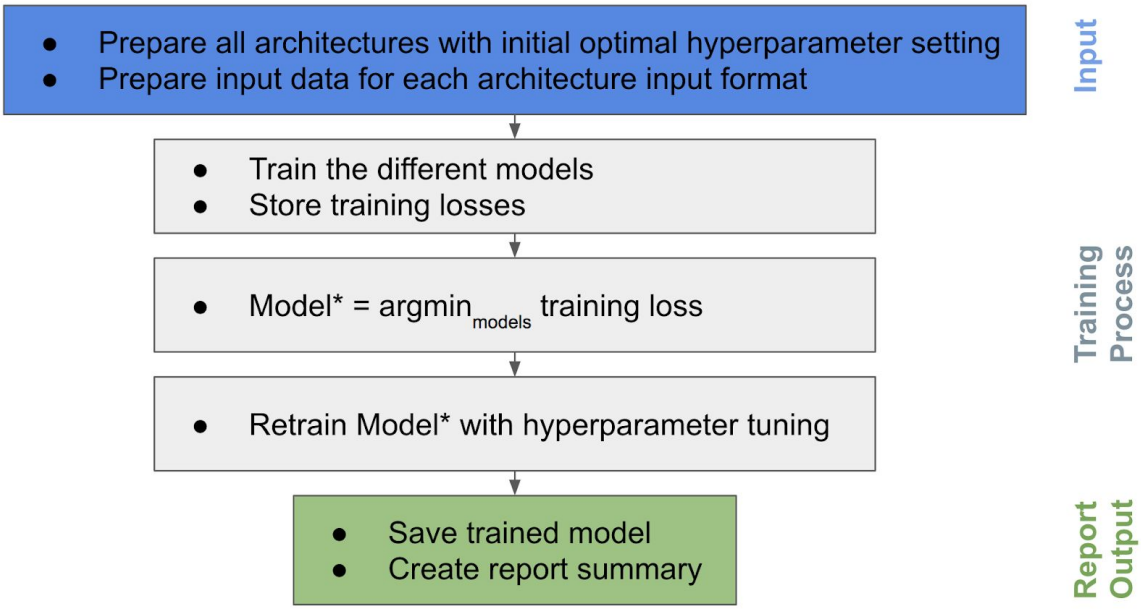
Fine Tuning:

We leverage the pre-training on ImageNet for each CNN that is available in the Keras API. All the models already have had training on a sufficiently diverse variety of objects to allow for transfer learning. Once the top performing model has been identified, we finetune the model by training only the last layers of the network. The prediction layer is also modified to have the same number of units as there are labels in the dataset.

Hyperparameter Tuning:

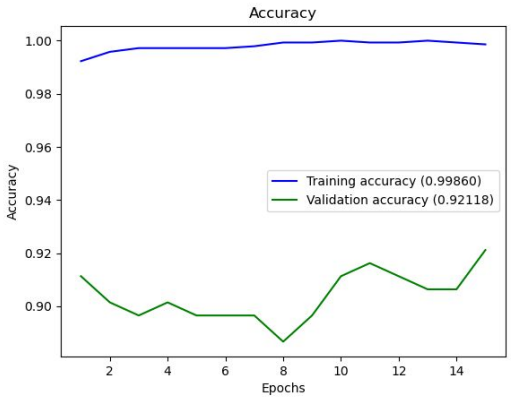
During the final round of training, the hyperparameters, such as learning rate and batch size, will be modified in order to find a higher accuracy for the model. By default, we lower the learning rate and increase the batch size.

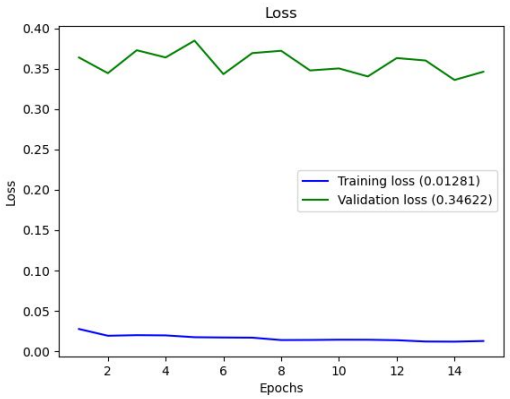
Training Workflow :

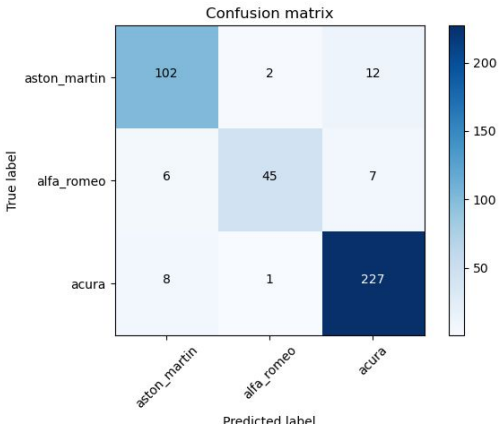


Results:

The service returns a report with training and validation results, along with a model architecture summary and link to the h5 model file that is stored in a Google Storage bucket. The following visuals provide an example of what the user will receive via email.







conv_pw_13 (Conv2D)	(None, 7, 7, 1024)	1048576
conv_pw_13_bn (BatchNormaliz	(None, 7, 7, 1024)	4096
conv_pw_13_relu (ReLU)	(None, 7, 7, 1024)	0
global_average_pooling2d (Gl	(None, 1024)	0
dense_1 (Dense)	(None, 3)	3075

Total params: 3,231,939
Trainable params: 1,053,699
Non-trainable params: 2,178,240

Conclusion/Discussion :

Given training time and cost constraints, we decided to come up with initial models and initial hyperparameters settings as optimal as possible . To do so, we performed an in-depth review of the literature to select the state of the art convolutional neural network architectures and investigated how transfer learning can be used on those architectures to further improve performances. Doing so, we were able to identify critical points (where to retrain and where to cut the pre existing architectures) in which transfer learning is useful i.e can learn general feature representation that can be transferred to any image dataset. In addition to that, it has been shown that a good hyperparameter initialization i.e learning rate, batch size , optimizer,... is critical to the network so we came up with a general suboptimal initialization scheme for those parameters . The process for choosing the best architecture is simple since we will just choose the architecture with the best performance and then fine-tune the hyperparameters of this specific architecture . In conclusion, this process is a simple end-to-end solution that is not costly and gives in general very good performances thanks to a good initialization scheme and the selection of the best architecture that is again fine tuned.

We are aware that hyperparameter optimization as well as architecture selection is an art and could be further improved however given cost constraints on the Cloud we decided to come up with a simple solution that is at no cost to the user and that he/she can use right away in addition to a report summarizing the training and validation metrics.

Future works in this cloud end-to-end service will include a paid service with a cost limit so that the user can have a more optimized neural network . To do so, we will use genetic algorithms for hyperparameter optimization and similarity techniques (P2L) to find the most appropriate architecture and dataset for transfer learning.

References :

François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. arXiv:1610.02357, 2017.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich. Going deeper with convolutions. arXiv:1409.4842, 2014.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. arXiv:1512.03385, 2015.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, Qing He. A Comprehensive Survey on Transfer Learning. arXiv:1911.02685, 2019.

Bishwaranjan Bhattacharjee, John R. Kender, Matthew Hill, Parijat Dube, Siyu Huo, Michael R. Glass, Brian Belgodere, Sharath Pankanti, Noel Codella, Patrick Watson. P2L: Predicting Transfer Learning for Images and Semantic Relations. arXiv:1908.07630, 2019.

Xueli Xiao, Ming Yan, Sunitha Basodi, Chunyan Ji, Yi Pan. Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm. arXiv:2006.12703, 2020.

David Laredo, Yulin Qin, Oliver Schutze, Jian-Qiao Sun. Automatic Model Selection for Neural Networks. arXiv:1905.06010, 2019.

Google, Inc. (2020, Nov, 16). What Is Cloud Storage?. Google. https://cloud.google.com/storage/docs/introduction

Google, Inc. (2020, Dec, 9). Build your deep learning project fast on Google Cloud. Google. https://cloud.google.com/deep-learning-vm

Mansouri, I. (2019, Mar, 27). Computer Vision Part 4: An overview of Image Classification architectures. Medium. https://medium.com/@ilias_mansouri/part-4-image-classification-9a8bc9310891

Karim, R. (2019, Jul, 29). A compiled visualisation of the common convolutional neural networks. Towards Data Science. https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

Szegedy et al. (2016, Feb, 23). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. Papers With Code. https://paperswithcode.com/method/inception-resnet-v2-a

Culurciello, E. (2017, Mar, 23). Neural Network Architectures. Medium. https://towardsdatascience.com/neural-network-architectures-156e5bad51ba

Bokka, K. (2019, Feb, 18). Guide to choosing Hyperparameters for your Neural Networks. Towards Data Science. https://towardsdatascience.com/guide-to-choosing-hyperparameters-for-your-neural-networks-38244e87dafe

Brownlee, J. (2016, Aug, 9). How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras. Machine Learning Mastery. https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/