Selection and Optimization of Deep Convolutional Neural Networks as a Service

Yanis Tazi
Erik Weissenborn

NYU

Executive summary

- End to end cloud service to identify a suboptimal model with hyperparameter selection for any given image classification task

- Build a quick, efficient neural network model selection and training process on Google Cloud with a built-in report summary directly sent to the user and ready to use

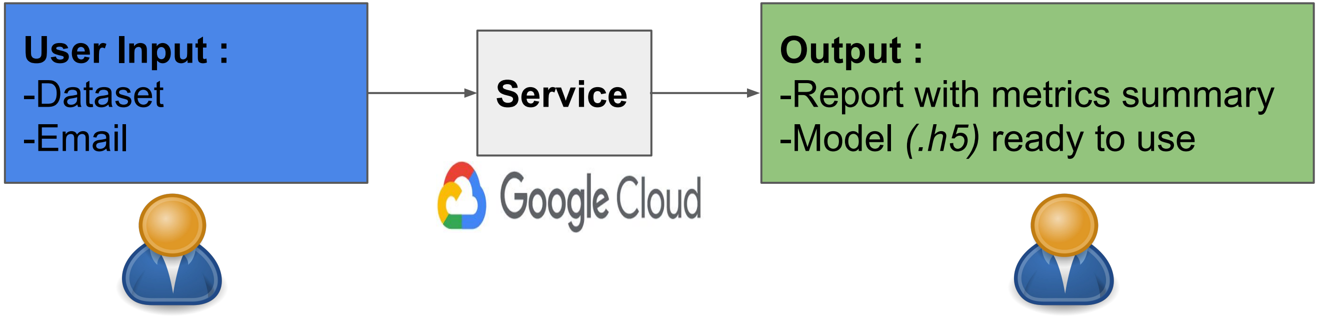
- Delivery solution with report summary of the selected model and metrics summary

- Contribute to the Deep Learning community by providing a user-friendly service handling training as well as providing GPUs on a cloud based approach

Problem Motivation

- Model selection and hyperparameter optimization is an art and can not be learnt in books.
- Computational resources (GPUs, memory) and technical skills for deep learning framework use and cloud implementations.

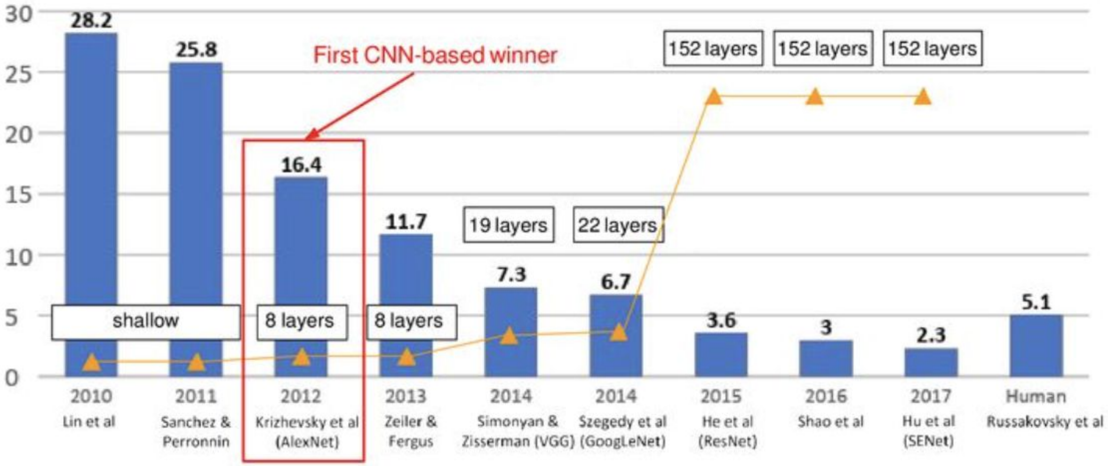
→ End to end cloud service with a report summary and an optimal finetuned model ready to use



Background / Technical challenges

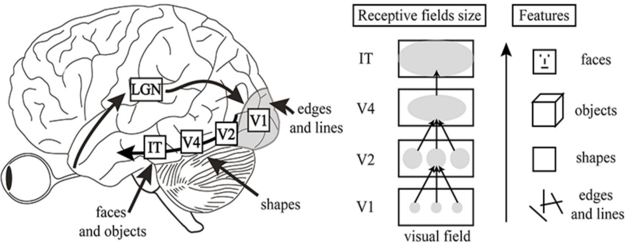
Why CNN ?

- Significant improvement of classification performances
- Automatic detection of important features and the network learns its own feature representation
- Biological motivations



Feature engineering

No Feature engineering



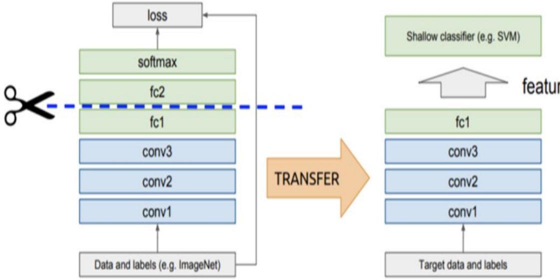
General architecture done (CNN)
What else can we optimize?

Our goal : build an efficient model with as little training as possible (time and cost constraints) → come up with the most optimized model pre training

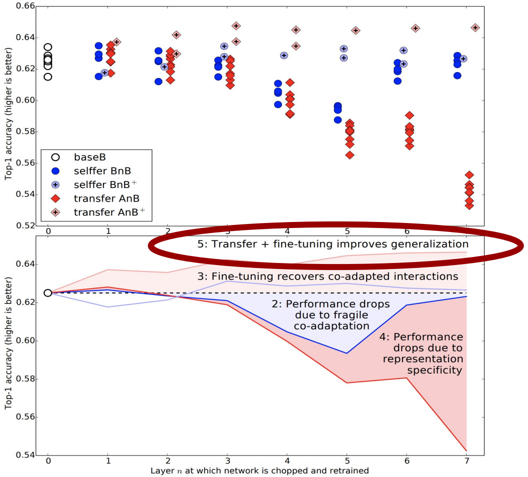
- Model choice ✔️

What else can we do efficiently to improve performances ?

- Transfer learning :

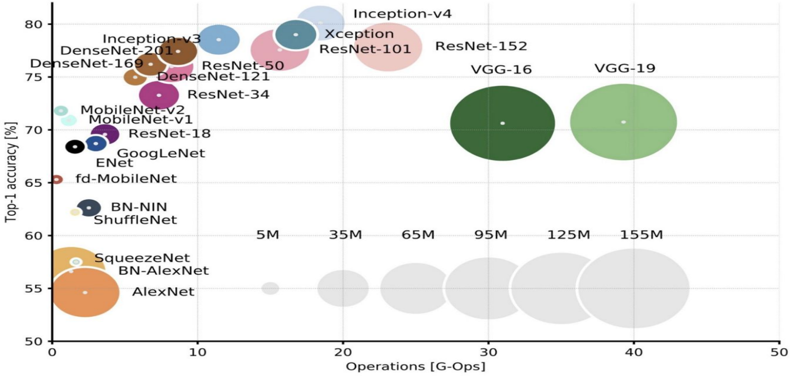


Transfer Learning with Pre-trained Deep Learning Models as Feature Extractors



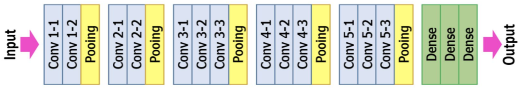
How transferable are features in deep neural networks? Yosinski et al.

- Architecture performances:

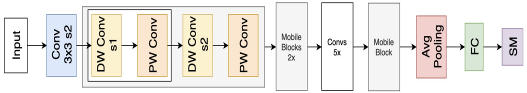


- Architecture choice

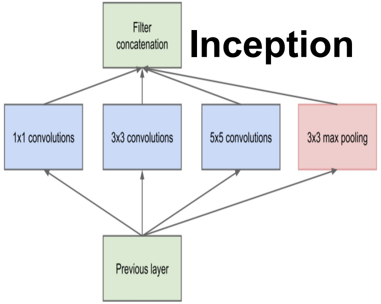
VGG-16



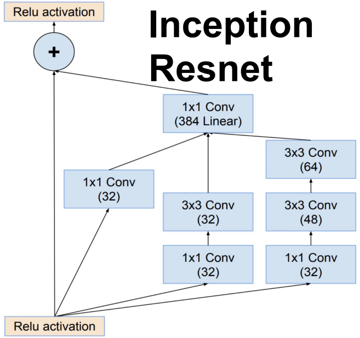
MobileNet



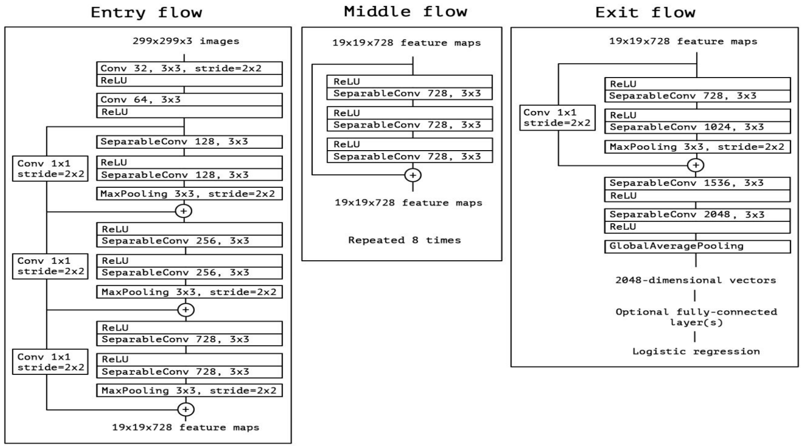
Inception



Inception Resnet



Xception



- Most important hyperparameter to optimize :
  - Number of layers to freeze during training
  - Number of layers to add / remove
  - Learning rate
  - Batch size
  - Optimizer
- Main challenge : exponential growth and lack of boundaries for hyperparameter search

Approach

- In depth review of convolutional neural network literature:
  - → Identification of the state of the art architecture to initialise the model with : number of layers to freeze / add for those selected pretrained models
  - Hyperparameter optimization with optimal initial configuration to limit hyperparameter search

- Initial training for the different architecture and performance comparison
  - → selection of best architecture
    - Retraining + finetuning of hyperparameters

Diagram

- Prepare all architectures with initial optimal hyperparameter setting
- Prepare input data for each architecture input format

Input

- Train the different models
- Store training losses

- Model* = $\text{argmin}_{\text{models}}$ training loss

- Retrain Model* with hyperparameter tuning

Training Process

- Save trained model
- Create report summary

Report Output

Implementation Details

1. Setup a Google Cloud Function for HTTP endpoint.
2. Allocate a Google Compute Deep Learning VM with NVIDIA K80 GPU.
3. Create Google Storage Buckets for receiving user datasets, user email addresses, and storing saved models.
4. Use Tensorflow/Keras framework to code an automatic Convolutional Neural Network model search and hyperparameter optimization process.
5. Setup an email account used for sending model reports on behalf of service.

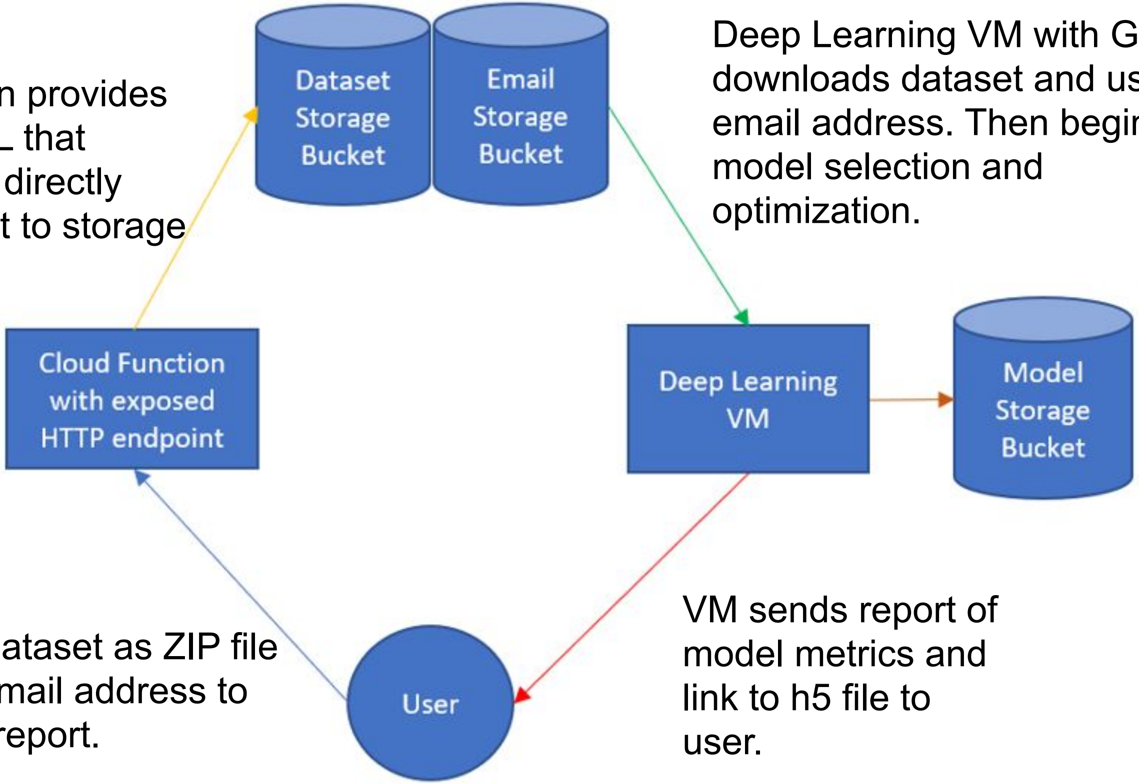
Google Cloud Functions

Google Compute Engine

Google Cloud Storage

Cloud Function provides presigned URL that allows user to directly upload dataset to storage bucket.

Deep Learning VM with GPU downloads dataset and user email address. Then begins model selection and optimization.

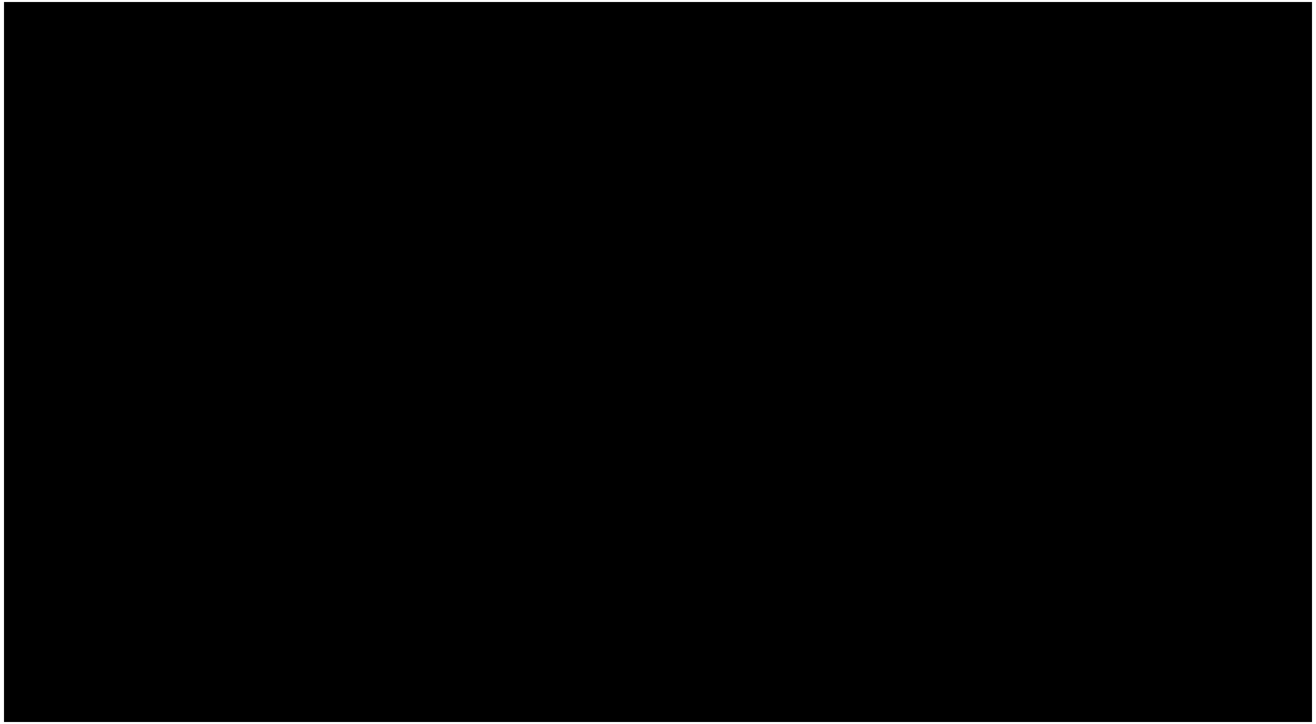


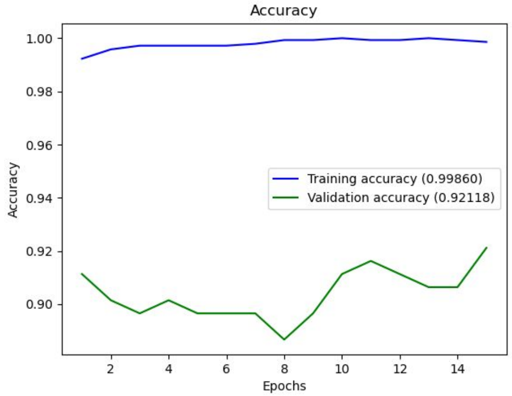
Model with best validation accuracy saved in storage bucket.

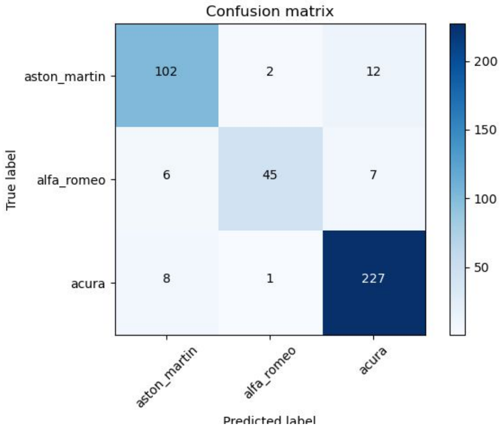
User uploads dataset as ZIP file and provides email address to receive model report.

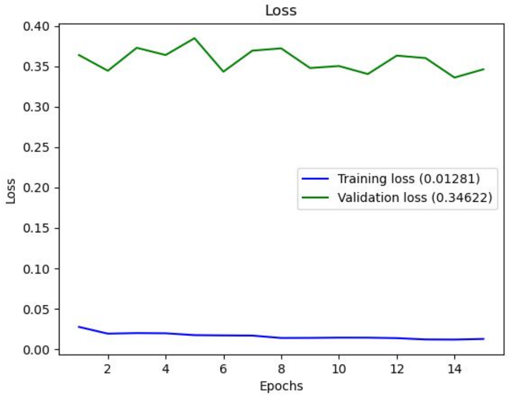
VM sends report of model metrics and link to h5 file to user.

Demo










```
_________________________________________________________________
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
_________________________________________________________________
```

The goal of providing a service that would deliver a trained and fine tuned CNN along with performance metrics to a public user was achieved. Our service is reasonably flexible, it can process unique datasets that contain classes and images of varying cardinality.

Conclusion

Although the service we have built does not completely explore the entire Neural Architecture Searchspace nor does it perform complex hyperparameter configuration search, it meets the base requirements and goal that was established when beginning the experiment.
There is more possibility for expansion and improvement of the current service than what is currently implemented. The current service can be seen a foundation for more intelligent future iteration that would include the following:

- More complex model architecture search, which may entail multi-objective optimization such as highest accuracy & lowest number of parameters.
- Hyperparameter optimization via published algorithms such as Hyperband

Extensions - Future work

- Add a cost service where user can specify how much they are willing to pay for this end to end service to get the most efficient ready to use neural net model. More resource intensive approach may be taken for a higher fee.
- P2L : find the most similar neural network architecture with the most similar dataset to transfer on.
- Efficient hyperparameter search : genetic algorithms for hyperparameter optimization.

- Increase flexibility and robustness of service by using a containerized pipeline approach to develop and deploy the workflow.
Would allow for easy experimentation and implementation of various training strategies for different dataset types.

"Overview of Kubeflow Pipelines," 27 Nov 2020 https://www.kubeflow.org/

The following GitHub repo contains a comprehensive README file and all of the code written for the model architecture search, hyperparameter optimization, and interfacing with Google Storage.

https://github.com/yanistazi/DeepCNN-as-a-Service