Machine Learning Engineer Nanodegree

Capstone Proposal

AnnMargaret Tutu

April 27th, 2020

Domain Background

Novel coronavirus (COVID-19) has brought to light many bottlenecks and subsequently areas for improvement regarding the medical supply chain. Simply put, testing labs cannot keep up with this level of demand. A recent article published by McKinsey **[2]** calls for making the medical supply chain more agile. More specifically, the article advises “digitalizing the medical supply chain.”

Digitalizing testing seems to be a good place to start to that effect, as serology, or the reverse transcription of polymerase chain reaction (rPT-PCR) **[3]**, revolves around a testing kit ecosystem of brand-specific instrumentation (e.g., swabs, reagent, etc.) that deplete at a rapid rate. Further, lab testing can take days to process, is prone to contamination **[1]** and requires confirmation phases to minimize the number of false positives/negatives. Even with ramped up capacity for rPT-PCR testing, the accuracy of these tests leave room for improvement. The need for alternative solutions in preparation for seasons where multiple viral infections circulate simultaneously will help to lessen the burden on ICUs.

Researchers and scientists provide a growing body of literature to support the value proposition of using radiographic imaging more robustly to pre-screen upper respiratory infectious diseases at scale. Together with advances in artificial intelligence (AI) **[8]** — and specifically, machine and deep learning along with transfer learning (TL) using pre-trained deep Convolutional Neural Networks (CNNs) — high-performance, machine aided chest X Ray (CXR) image classification demonstrates a less error-prone solution, for less money, with existing infrastructure in place at most hospitals and medical facilities.

Problem Statement

In their paper, **[4]** present a convolutional neural network (CNN) architecture called *COVIDNet*, capable of differentiating between “normal” CXR images and CXRs with either Pneumonia or COVID-19 present. The authors illustrate the network architecture with the following visual:

Machine Learning Engineer Nanodegree

Capstone Proposal

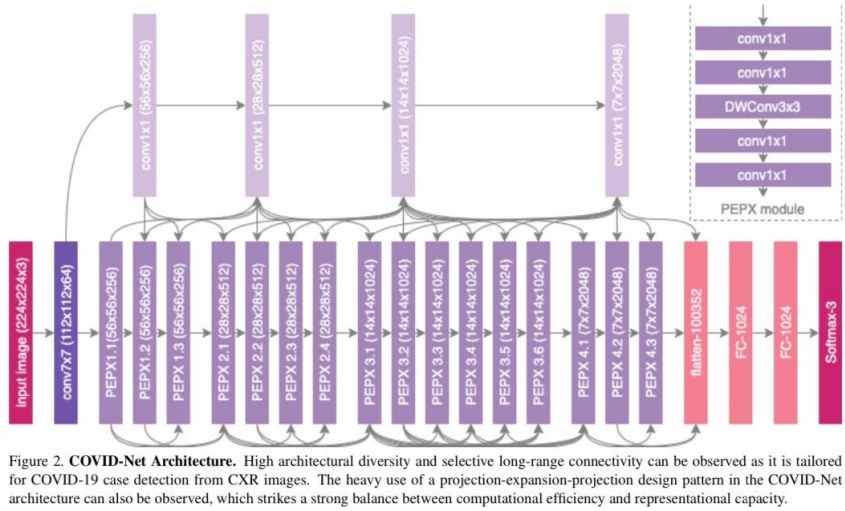


Figure 2. **COVID-Net Architecture.** High architectural diversity and selective long-range connectivity can be observed as it is tailored for COVID-19 case detection from CXR images. The heavy use of a projection-expansion-projection design pattern in the COVID-Net architecture can also be observed, which strikes a strong balance between computational efficiency and representational capacity.

COVID-Net Illustration (L. Wang and A. Wong., 2020)

The network gains increased representational capacity for the COVID-19 classification task from the design pattern used to implement the core module, the PEPX unit — a light-weight residual unit. These units are stacked successively in blocks increasing in size exponentially, each block followed with a proceeding pooling layer for increased representational capacity.

Datasets and Inputs

COVIDx Open Data Set

Authors of the paper have also open-sourced the samples used to train and test the model. The project provides documentation explaining the data wrangling process and provides a template notebook to preprocess the data (e.g., generating 2D .jpg images from 3D CXR .dem files; a major contribution to the research community, as COVID-19-related imagery is scarce. Roughly ⅓ of the data is openly available on git and ready for download. The other ⅔ of the data is available via Kaggle competition page for the RSNA Pneumonia Detection Challenge. The data distribution is as follows:

Split	COVID-19*	Pneumonia	Normal	Total
Training	152	5451	7966	13569
Testing	31	100	100	231

Instructions for accessing the data and .dem to .jpg processing steps are available on the readme page for the COVID-Net project on GitHub.

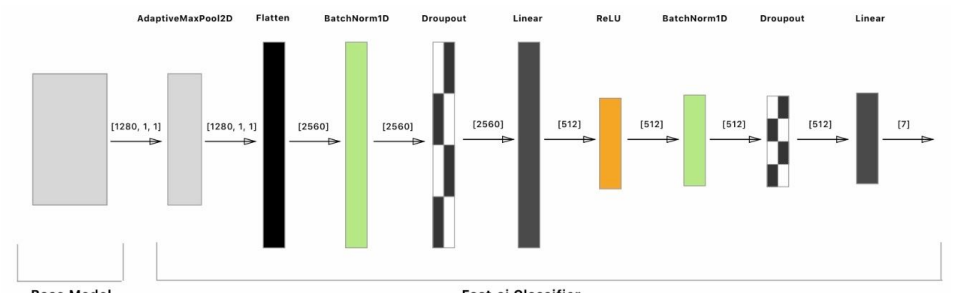
***Note:** Due to class imbalances (e.g., far fewer COVID-19 examples) authors utilize a class rebalancing strategy at the batch level, but no not go into detail.

Machine Learning Engineer Nanodegree

Capstone Proposal

Solution Statement

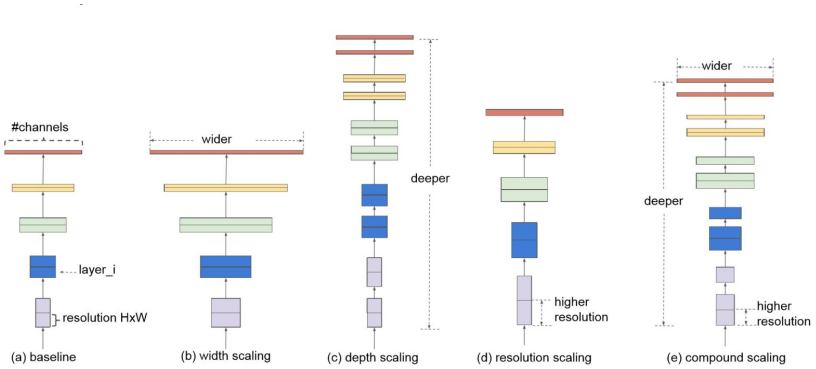
For this project, I propose that it may be possible to reach similar, if not better, results to **[4]** by leveraging a popular data science library built on top of PyTorch called *fast.ai* **[9]**. In essence, the fast.ai provides an intuitive set of utility methods informed by cutting-edge research to maximize results on training, specifically for small data. In addition, fast.ai replaces the linear classification layer of a passed in base architecture, with a classifier comprised of the following:



As I illustrate above, the fast.ai classifier is a module list that applies an AdaptiveMaxPool2D layer on final features before flattening to a rank-1 tensor and applying batch normalization, non-linearity and dropout to produce the final feature embedding.

Results from research presented in **[5]** achieved descent results compared to COVIDNet on the same dataset using far less computational resources; leveraging fast.ai, TL on a pretrained ResNet-50 base architecture, cyclical learning rates **[4]**, and human-intervention to implement a progressive resizing strategy.

My proposed solution takes the approach described in **[5]** but utilizes an alternative, more *scalable* base architecture to increase the capacity of the progressive resizing strategy with far less overhead, in terms of memory usage. The following illustration demonstrates how compound scaling works for *EfficientNet* architectures **[7]** (as these networks are called) in contrast to legacy networks, like ResNets.

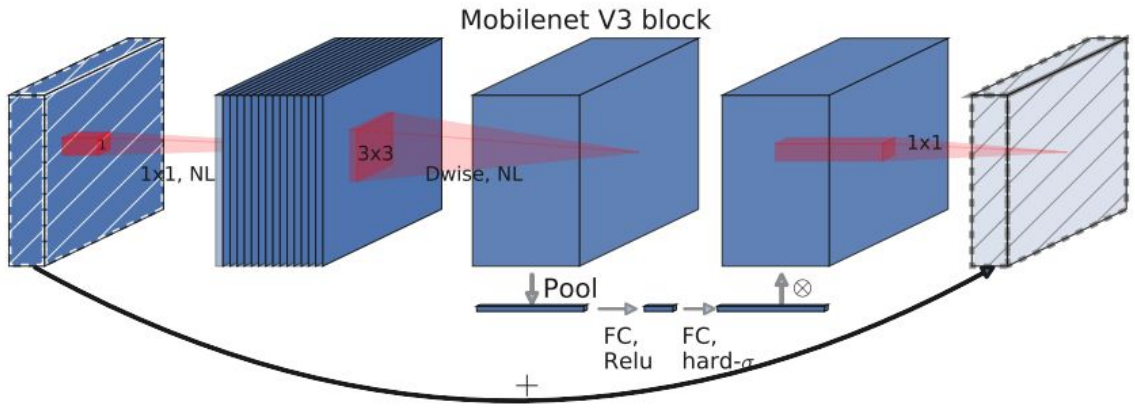


Compound Scaling Illustration (Tan and Le, 2019)

Machine Learning Engineer Nanodegree

Capstone Proposal

EfficientNets are mainly comprised of Mobile Inverted Bottleneck Blocks (**MBConv6**) blocks —first presented in this **paper** — with squeeze-and-excitation (SEB) layers added on occasion:



MobileNet v2 Block with SEB → MobileNetV3 (A. Howard et al., 2019)

Regular residual blocks use skip-connections to forward activation identity up the network with a wide-narrow-wide approach to channel layering, whereas MBConv6 blocks (inverted residual blocks) do the inverse and take a narrow-wide-narrow approach to channel layering, using SEBs. SEBs improve upon how residual blocks weight their channels (typically, weighting all channels equally) by injecting a mini-connected block/network that calculates a weight vector to model interchannel dependencies within the block. I found this post very helpful for unpacking the functionality of SEBs.

Note that **[5]** does not mention a sampling strategy to account for imbalances in the data (e.g., far fewer training examples for COVID-19 than the two other classes) but **[4]** does. For this reason, I opted out of including results from **[5]** in the benchmark model detailed in the following section. Similar to **[4]**, the proposed solution will also use a weighted sampler at the batch level to strategically oversample the underrepresented class.

In summary, the goal of this project is as follows:

- Download, label and preprocess the COVIDx Open Dataset
- Train a CXR image classifier/estimator — capable of differentiating between 3 categorical states of infection for the upper respiratory system — using an AWS SageMaker provisioned runtime
- Deploy a predictor for this task to a public API endpoint exposed by AWS Gateway in the form of an AWS Lambda Function
- Design/develop a web interface for this inference service that enables end users to access the API

Machine Learning Engineer Nanodegree

Capstone Proposal

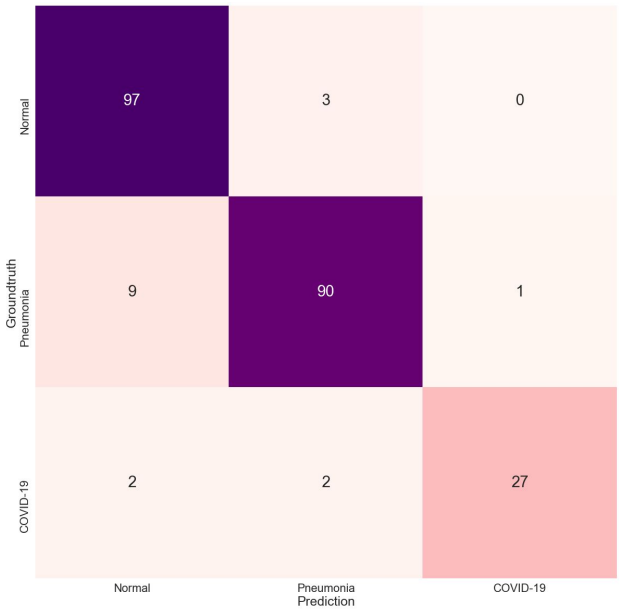
Benchmark Model

The experiment design described in the paper involves pretraining the architecture on ImageNet and performing transfer learning (TL) with the following specifications:

Hyperparameters	Transformations
learning rate=2e-5 **epochs**=22 **batch size**=8 **factor**=0.7 **patience**=5.*	Translation Rotation Horizontal Flip Intensity Shift.

***Note**: In the paper, authors briefly mention a “patience policy” where learning rate decreases as learning stagnates. More on this learning schedule can be found here in the PyTorch docs.

The research reports the following results for the **COVIDNet-CXR Small** (the shallower variant of the model):



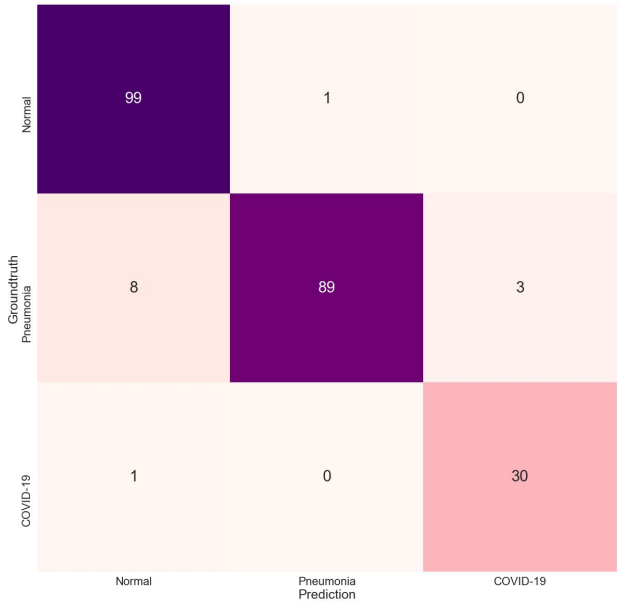
Sensitivity (%)		
Normal	Pneumonia	COVID-19
97.0	90.0	87.1

Machine Learning Engineer Nanodegree

Capstone Proposal

Positive Predictive Value (%)		
Normal	Pneumonia	COVID-19
89.8	94.7	96.4

Results for the **COVIDNet-CXR Large** (the deeper variant of the model):



Sensitivity (%)		
Normal	Pneumonia	COVID-19
99.0	89.0	96.8

Positive Predictive Value (%)		
Normal	Pneumonia	COVID-19
91.7	98.9	90.9

This historical model will serve as a basis for comparison.

Machine Learning Engineer Nanodegree

Capstone Proposal

Evaluation Model

To evaluate performance on the validation set, and give special attention to minimizing false negatives, I'll be aggregating the following **metrics** on a batch-by-batch basis:

- Accuracy → (TP+TN)/(TP+FP+FN+TN)
- Precision/PPV → TP/(TP+FP)
- Recall/Sensitivity → TP/(TP+FN)
- F-score → 2*(Recall * Precision) / (Recall + Precision)
- Specificity → TN/(TN+FP)

Where:

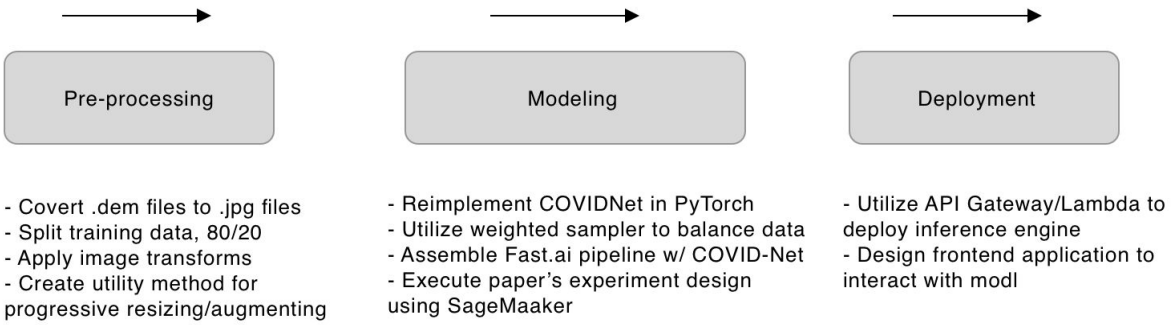
TP → True Positives
TN → True Negatives
FP → False Positives
FN → False Negatives

Additionally, following training and before deployment, I will perform a quick inference test with the provided labeled test set, Fast.ai simplifies the process of applying Test Time Augmentations (TTA) and calculating metrics for test data. This article helped me wrap my head around clinical metrics.

I think these metrics are appropriate to maximize, as we are modeling to reduce infection spread for scenarios in which there is high prevalence.

Project Design

The project design follows the MLE workflow (e.g., pre-process, model, deploy) expounded upon in Lesson 3 modules:



Beneath each axiom of the workflow, specific steps are listed to provide a clear path forward for this project.

Machine Learning Engineer Nanodegree

Capstone Proposal

References:

[1] D. Willman, “Contamination at CDC lab delayed rollout of coronavirus tests.” Apr. 18, 2020. [Online] Available: https://www.washingtonpost.com/investigations/contamination-at-cdc-lab-delayed-rollout-of-coronavirus-tests/2020/04/18/fd7d3824-7139-11ea-aa80-c2470c6b2034_story.html

[2] E. Barriball et al., “Supply-chain recovery in coronavirus times—plan for now and the future.” Mar. 2020. [Online]. Available: https://www.mckinsey.com/business-functions/operations/our-insights/supply-chain-recovery-in-coronavirus-times-plan-for-now-and-the-future

[3] K. Green et al., “What tests could potentially be used for the screening, diagnosis and monitoring of COVID-19 and what are their advantages and disadvantages?” CEBM. Apr. 20, 2020 [Online]. Available: https://www.cebm.net/wp-content/uploads/2020/04/CurrentCOVIDTests_descriptions-FINAL.pdf

[4] L. Wang and A. Wong, “COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID19 Cases from Chest Radiography Images,” ArXiv200309871 Cs Eess, Mar. 2020 [Online]. Available: http://arxiv.org/abs/2003.09871.

[5] L. N. Smith, “Cyclical Learning Rates for Training Neural Networks,” ArXiv150601186 Cs, Apr. 2017. [Online]. Available: http://arxiv.org/abs/1506.01186.

[6] M. Farooq, A. Hafeez, “COVID-ResNet: A Deep Learning Framework for Screening of COVID19 from Radiograph,” arXiv:2003.14395, 2020. Available: https://arxiv.org/abs/2003.14395

[7] M. Tan and Q. Lee, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” arXiv:1905.11946, Nov. 23, 2019. Available: https://arxiv.org/pdf/1905.11946.pdf

[8] O. Gozes et al., “Rapid AI Development Cycle for the Coronavirus (COVID-19) Pandemic: Initial Results for Automated Detection & Patient Monitoring using Deep Learning CT Image Analysis,” ArXiv200305037 Cs, Mar. 2020. [Online]. Available: http://arxiv.org/abs/2003.05037.

[9] J. Howard and S. Gugger, “fastai: A Layered API for Deep Learning,” Information, vol. 11, no. 2, p. 108, Feb. 2020, doi: 10.3390/info11020108.