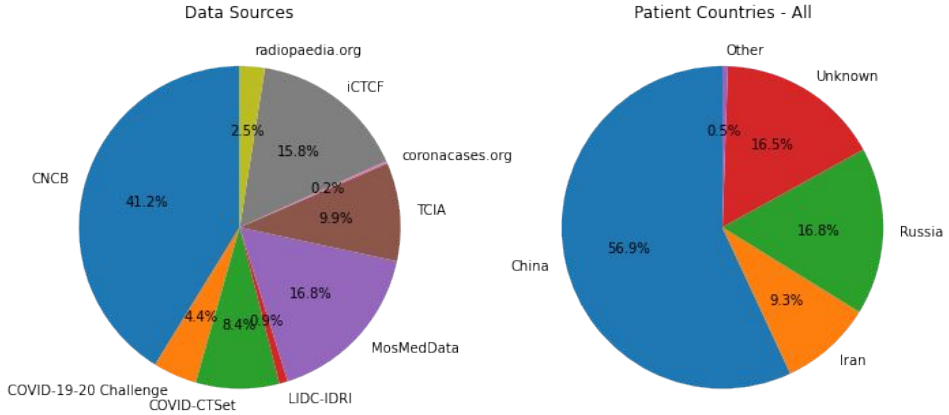
Diagnosing COVID Infection Using Chest CT Scans and Deep Learning Models

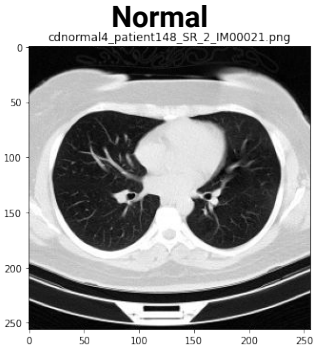
Simon Li, Matthew Hui, David Yen, Jason Papale

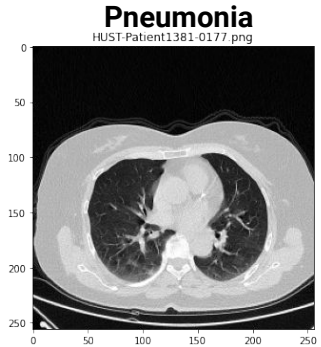
Project Overview

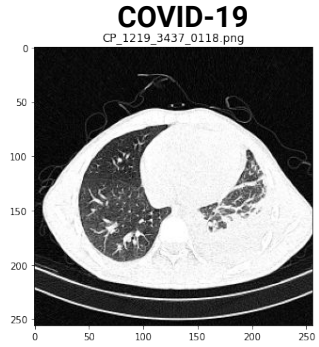
Goals	Project Components
● Understand performance of COVID-19 detection using deep learning ● Demonstrate effectiveness of existing deep CNNs in a diagnostic setting ● Simulate usage at point of care	● Develop baseline CNN ● Augment data using GAN ● Apply transfer learning using existing learning architectures ● Implement on an edge device

COVIDx CT-2 Data Set



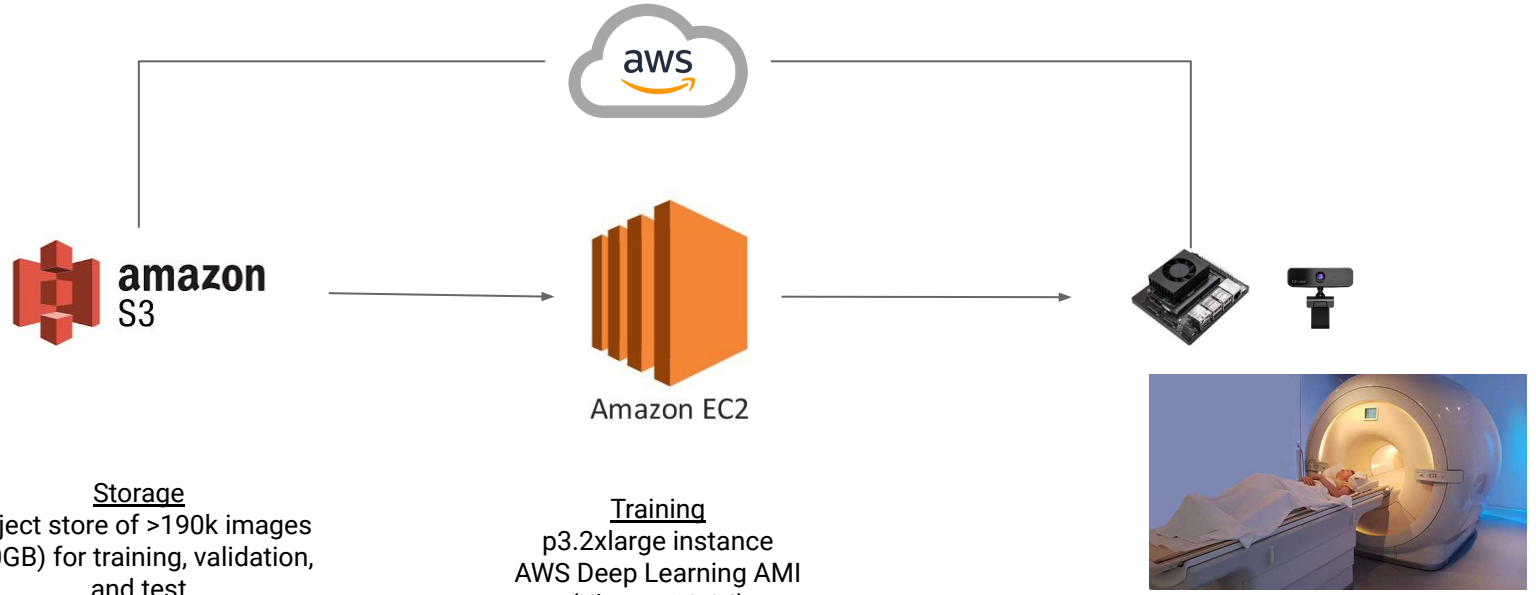






- Open source dataset of ~195,000 CT images from 3745 patients across the globe
- 3 Classes: Normal, Non-Covid-19 Pneumonia, COVID-19

Pipeline



Storage
Object store of >190k images (30GB) for training, validation, and test

Secure storage of classified images

Training
p3.2xlarge instance
AWS Deep Learning AMI (Ubuntu 18.04)
Pytorch

Inference
Docker container:
Nvidia L4T-ML, PyTorch, Torchvision, AWS

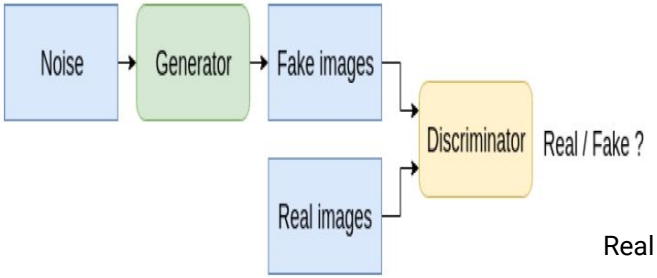
Conditional GAN Architecture for CT Images

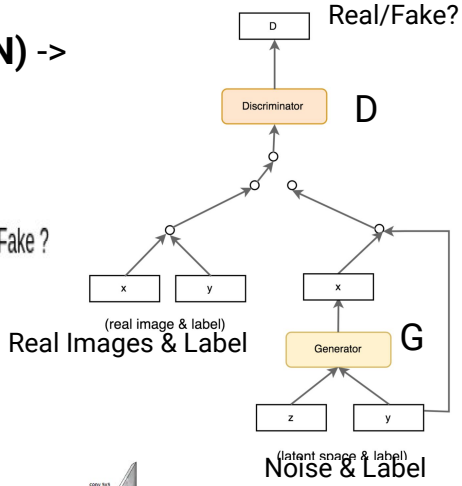
Yann LeCun: “GAN is the most interesting idea in Machine Learning in last 10 years”

cGAN Architecture features:

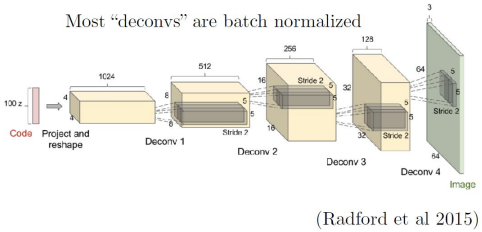
- 100 latent space dimensions
- 50 class label embedding
- 5 deconv/conv layers with 2x scaling
  - 8x8x512 - 16x16x256
  - 16x16x256 - 32x32x128
  - 32x32x128 - 64x64x64
  - 64x64x64 - 128x128x32
  - 128x128x32 - 256x256x16
- 5x5 kernel/stride 2 for all hidden layers
- Leaky ReLU for all hidden layers
- Batch Normalization (optional)
- Tanh/Sigmoid for G/D out-layer
- Binary cross entropy loss function
- ADAM optimizer

GAN versus **conditional GAN (cGAN)** ->

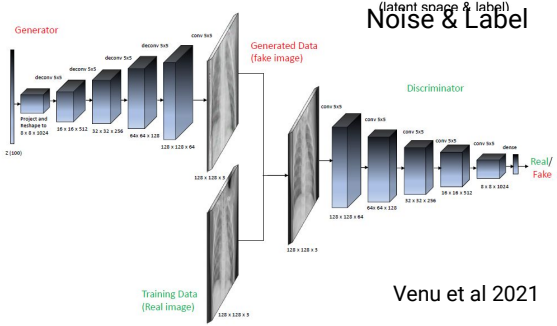




DCGAN Architecture



(Radford et al 2015)

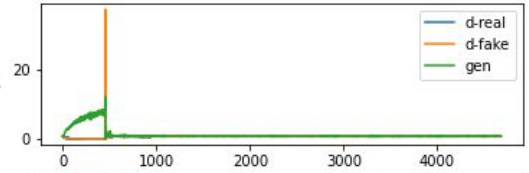


Venu et al 2021

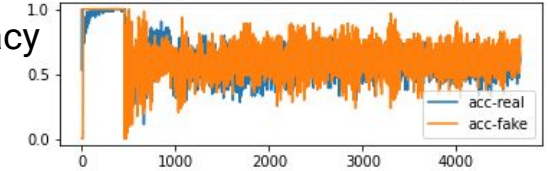
cGAN Training Results

Baseline training of 28x28 resolution images

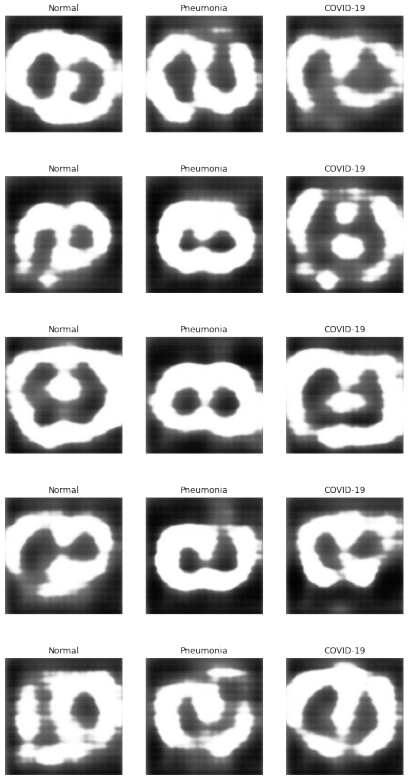
loss

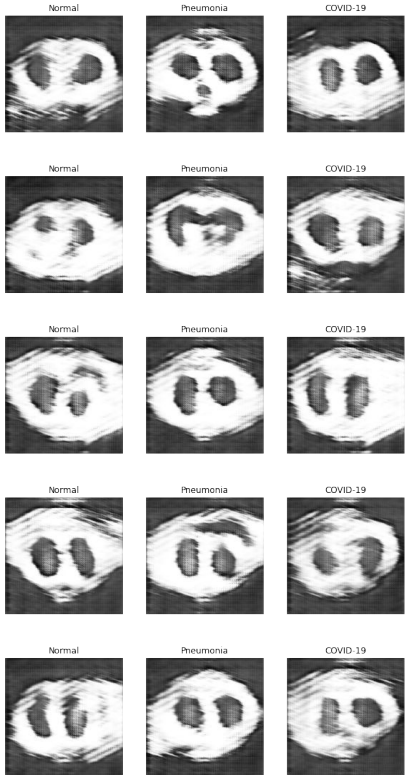


accuracy



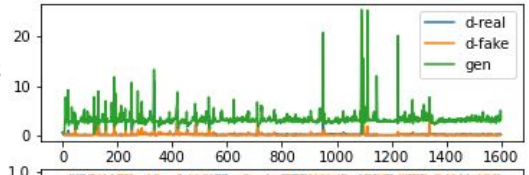
Sample cGAN images @100 epochs



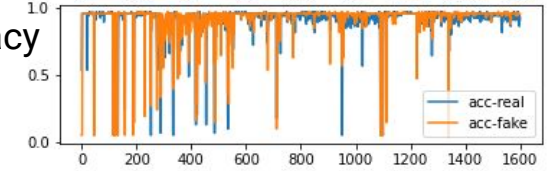


100 epochs training of 2k 256x256 images

loss



accuracy



Sample cGAN images @30 epochs

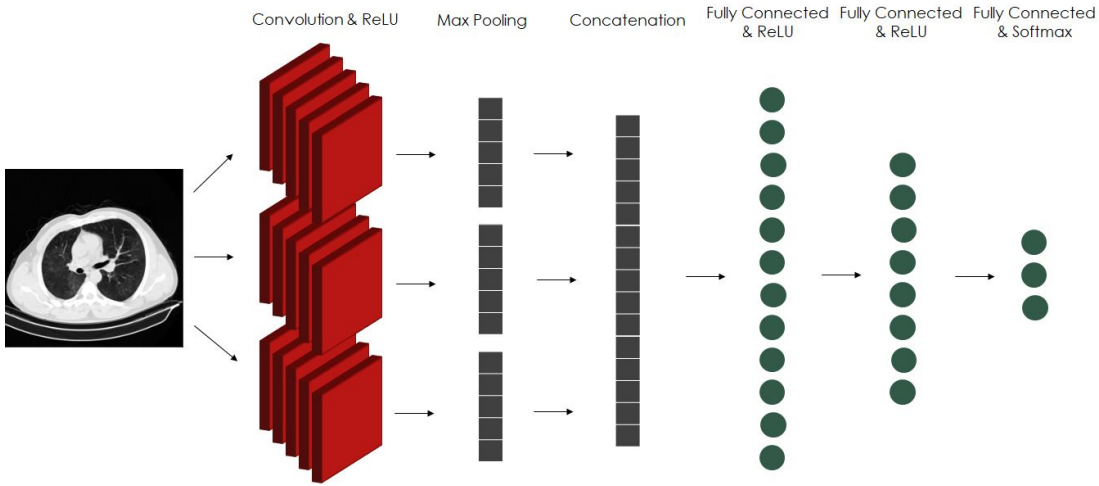
Baseline CNN – Architecture

Developed in Pytorch

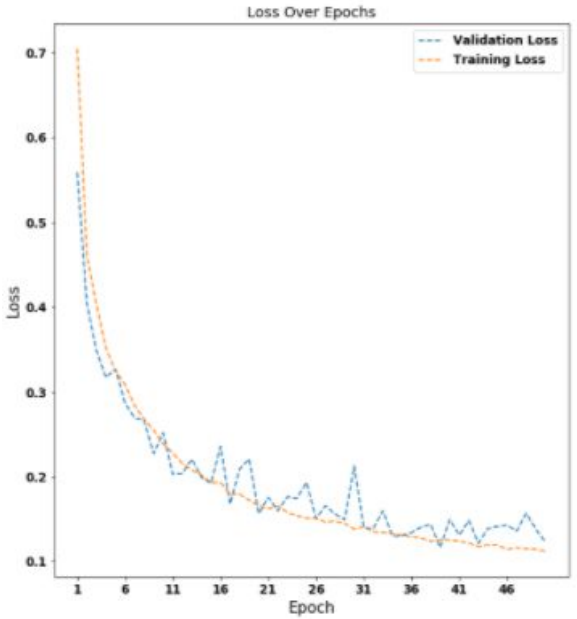
Hyperparameters selected after evaluation model performance on 72 possible combinations

Final model design included:

- 3 convolution & ReLU layers
  - 25 filters each
  - Kernel sizes of 2, 3, and 4
  - 2 fully connected layers
    - 200 neurons
    - 50 neurons
  - Output layer with Softmax

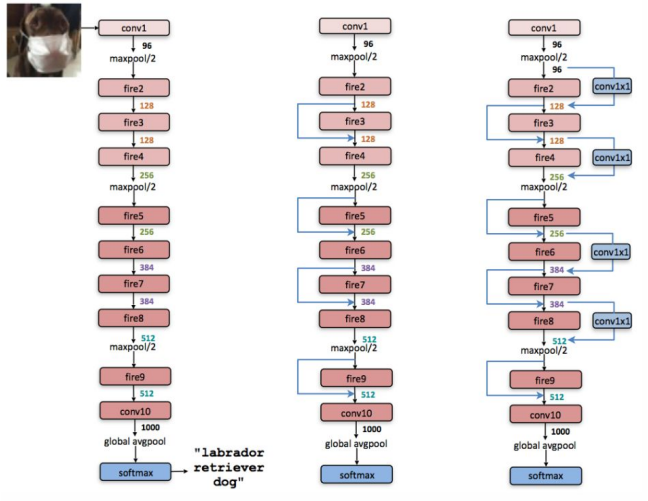


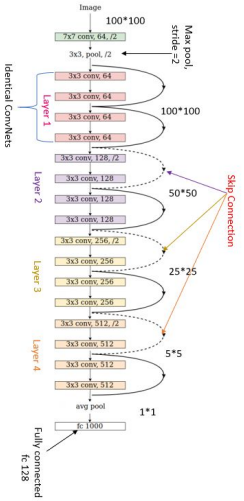
Baseline CNN – Training and Results



	precision	recall	f1-score	support
0	0.91	0.98	0.94	6032
1	0.95	0.94	0.94	4027
2	1.00	0.95	0.97	9433
accuracy			0.96	19492
macro avg	0.95	0.96	0.95	19492
weighted avg	0.96	0.96	0.96	19492

PreTrained Models - Architecture





Squeezenet: Compact replacement for Alexnet with 50x fewer parameters. Smaller size allows for easier deployment

Contains “Squeeze” (1x1 conv. filters) and “Expand” (1x1 and 3x3 conv. filters) layers and no fully connected layers

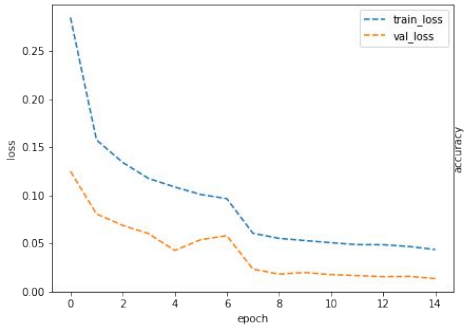
ResNet-18: uses skip connections in residual blocks that allow for training deep nets without vanishing/exploding gradients.

Pretrained on over 1MM images from ImageNet. Weights in each layer allowed to update with CT images

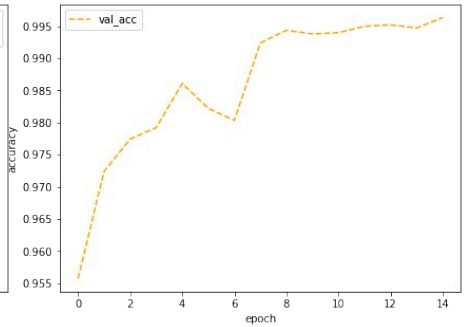
PreTrained Models - Training and Results

SqueezeNet

Loss over Epochs



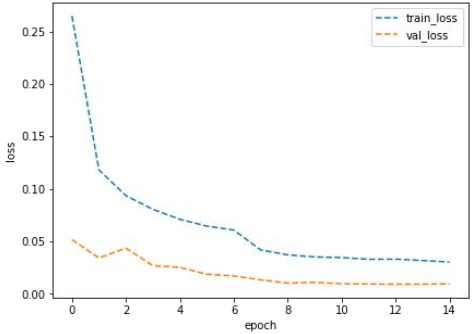
Val Accuracy over Epochs



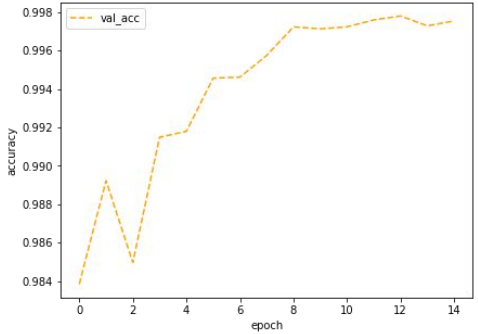
	precision	recall	f1-score	support
Covid	0.986	0.979	0.983	9433
Normal	0.980	0.990	0.985	6031
Pneumonia	0.966	0.967	0.966	4027
accuracy			0.980	19491
macro avg	0.977	0.979	0.978	19491
weighted avg	0.980	0.980	0.980	19491

ResNet-18

Loss over Epochs

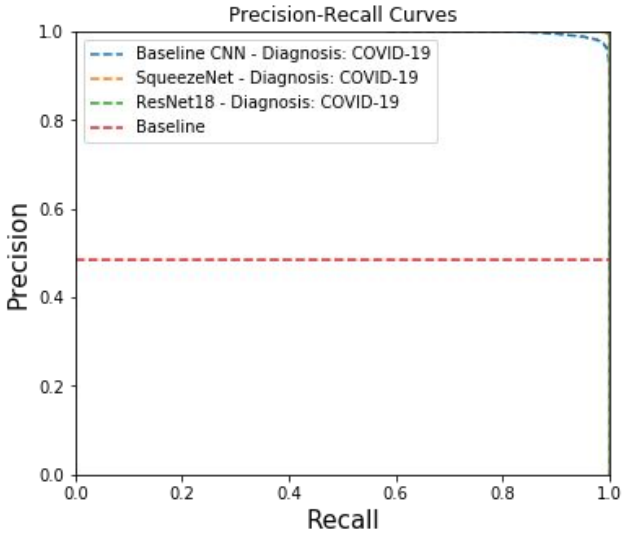


Val Accuracy over Epochs



	precision	recall	f1-score	support
Covid	0.997	0.998	0.997	9336
Normal	0.998	0.998	0.998	6057
Pneumonia	0.997	0.995	0.996	4098
accuracy			0.997	19491
macro avg	0.997	0.997	0.997	19491
weighted avg	0.997	0.997	0.997	19491

Model Comparison



Covid	Precision	Recall	F1 Score
Baseline CNN	1.000	0.950	0.970
Squeeze Net	0.986	0.979	0.983
ResNet-18	0.997	0.998	0.997

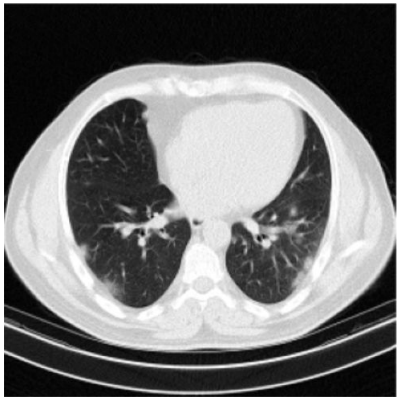
We were particularly interested in Recall and Precision - patients should not be sent home if they have Covid but were predicted Normal

On Covid classification, all three metrics. The added layers from Squeeze Net and ResNet-18 allowed them to outperform our baseline.

Deployment of Model to the NX

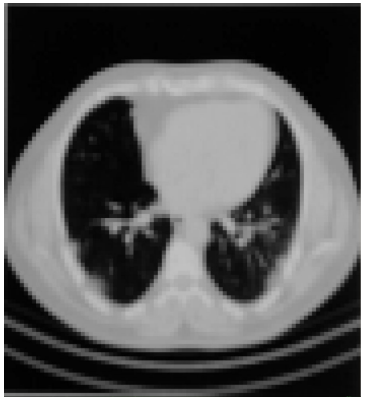
NX

Original CT Image
(Actual Diagnosis: COVID)



Snapshot of CT Image on Computer Screen Using Webcam

Cropped and Resized Image



```
predict_CNN(CNN_Model, COVID_Image)
```

```
Diagnosis: COVID-19 ; Probability of Diagnosis: 0.9075866148082662
```

```
results.....................................................
inference time: 4256 ms
Pneumonia : 0.730
results uploaded to s3.....................................
```

Conclusion

Developed an end-to-end pipeline for training and inference of Covid-19 CT scans.

Developed and evaluated three deep neural network models.

We also attempted to use GAN to generate images for scenarios where images of a new disease could be limited.

At inference, we deployed the model on the Jetson Xavier NX.

Questions?

Appendix

Background on GAN

History of GAN development

- 2014 Initial GAN Concept Proposed by Ian Goodfellow
- 2014 paper by Mirza, et. al. developed conditional GAN
- 2016 paper by Radford, et. al. incorporating Deep Neural Network in GAN
- 2016 paper by Chintala,, et. al. for tricks and hacks to stabilize GAN training
- 2018 paper by Tarras to extend GAN to high resolution images - Progressive GAN
- Proliferation of GANs in many applications: styleGAN, bigGAN, cycleGAN, self-attention conditional GAN, …

References:

1. Mirza, M. and Osindero, S., 2014, Conditional generative adversarial nets, arXiv:1411.1784
2. Radford, A., Metz, L. and Chintala, S., 2016, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, arXiv:1511.06434v2
3. Chintala, S.,, Denton, E., Arjovsky, M, and Mathieu, M., 2016, ganhacks: How to Train a GAN? Tips and tricks to make GANs work
4. Karras, T, Aila, T., Laine, S., and Lehtinen, J., 2018, Progressive Growing of GANs for Improved Quality, Stability and Variation, ICLR 2018
5. Arjovsky, M, Bottou, L., 2017, Towards Principled Methods for Training Generative Adversarial Networks, ICLR 2017
6. Teramoto, A, et.al., 2020, Deep learning approach to classification of lung cytological images: Two-step training using actual and synthesized images by progressive growing of generative adversarial networks, 10.1371/journal.pone.0229951

Proposed Approach for CT-Scan Application

Requirements:

- 256x256 high resolution images
- 3 classes - “Normal”, “Pneumonia”, “COVID-19”
- Generate desired number of realistic CT-Scan images for specific classes with good representation of diverse patient population

Proposed GAN Architecture and Design

- Conditional GAN to generate images of any specific class
- Use Deep Convolutional Neural Network architecture to cover high level structure of lung cross sections and low level detail of pathological features

GAN Model Development

- Baseline cGAN model architecture for 28x28 images with two conv/deconv layers for a factor of 4 magnification from 7x7x128 to 28x28x32
- Baseline cGAN model for resized CT image of 32x32 for 4x magnification
- Add matching network of one convolution layer in Discriminator and deconvolution layer in Generator for each increment of 2x resolution from 32x32 to 256x256
- Use GAN hack tricks to help stable GAN training (see the Best Practices and GAN-hackings Tricks slide)
- Experiment with model parameters
    - Try kernel sizes of 3x3, 4x4 and 5x5 for all conv/deconv layers and 8x8 for generator out layer
    - Try input images [1024, 2048], batch size [128, 256], number of epochs [10, 20, 30, 40, 50, 60, 100]
    - Add Batch Normalization for each conv/deconv layer (note: no help)
    - Try Wasserstein loss function to help training convergence (note: no help)

Best Practices and GAN-hacking tricks

Best Practices Used in DCGAN & cGAN

- Normalize input to [-1, 1]
- Add a embedding layer to class label inputs to divide them into 50 discrete bins
- Use a latent_dim of 100 for the generator
- Factor of 2 downsizing and upsizing using 5x5 kernel/stride 2 between layers in conv/deconv
- Use Leaky ReLU with alpha=0.2 as activation function for all conv/deconv layers
- Use Tanh as the activation function for Generator out-layer
- Use Sigmoid as the activation function for Discriminator and GAN model out-layer
- Use ADAM for optimizer with .0002 for learning rate and 0.5 for beta_1 regulation
- Use binary cross entropy as the loss function for both discriminator and GAN models
- Use a dropout of 0.4

General Recommendable GAN Hacks

- Use large kernels
- Use more filters
- Use stride rather than pooling for scaling conv/deconv layers
- Batch normalization
- Wasserstein loss function
- Noisy/soft label
- Separate training for real and fake images
- Gaussian weight initialization

GAN Results and Discussions

Training results (see slide 6):

- Loss/Accuracy plots versus training steps
- Sample generated images over epochs of training

Discussions, Lessons learned and future work

- GAN is difficult to train given known problems in stability, image quality (checkerboard artifacts) and diversity (mode collapse) due to the interplay (zero-sum game) of the discriminator and the generator during training
- Follow the best practice and GAN hack tricks could help
- Use proven architecture similar to the application will help
- Experimenting with new architectures and new insights could help develop new applications
- Require large training resources and time, so tricks that can reduce these are important
- Can be an important technique to protect privacy since generator never see input images and will not generate exact real images
- A possible future direction outside fine tuning the existing architecture is coupling of a high resolution progressive GAN with advanced CNN like VGG to resolve the image quality issue.