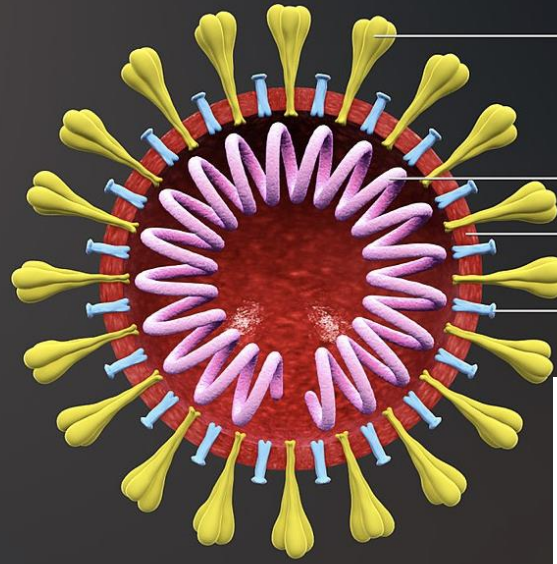
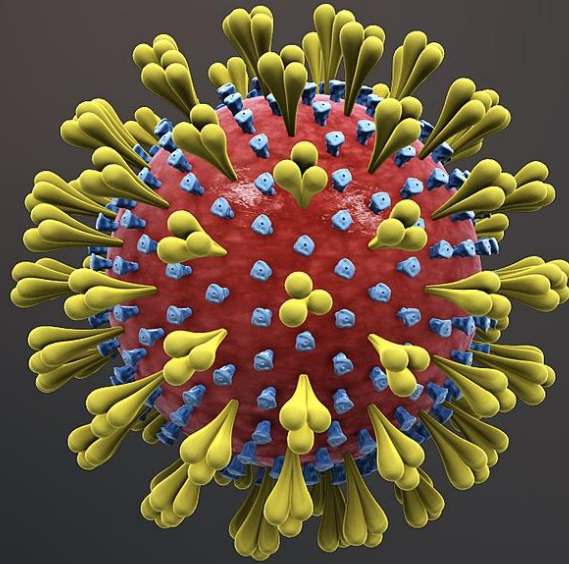


2019-NCOV CATEGORIZATION AND PREDICTION: A DEEP LEARNING ANALYSIS

Shuja Sajjad



2019'S NOVEL CORONAVIRUS



Spike Glycoprotein (S)

RNA and N protein

Envelope

Hemagglutinin-esterase dimer (HE)

- Is observed to have originated from China, with the first recorded cases in November 2019.
- The resulting illness is observed to have a strong **respiratory** component.

- Coronavirus (COVID-19) is an illness caused by a virus that can spread from person to person.
- The virus that causes COVID-19 is a new coronavirus that has spread throughout the world.
- COVID-19 symptoms can range from mild (or no symptoms) to severe illness.

Systemic:

- Fever
- Fatigue

Kidneys:

- Decreased function

Intestines:

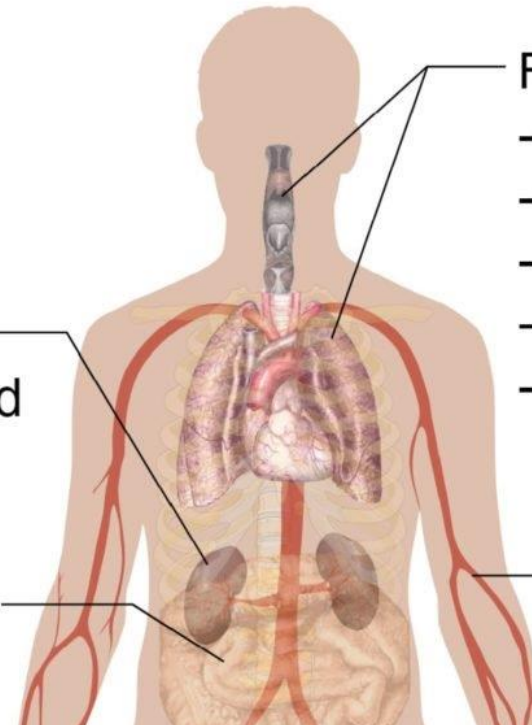
- Diarrhea

Respiratory:

- Sneezing
- Runny nose
- Sore throat
- Dry cough
- Shortness of breath

Circulatory system:

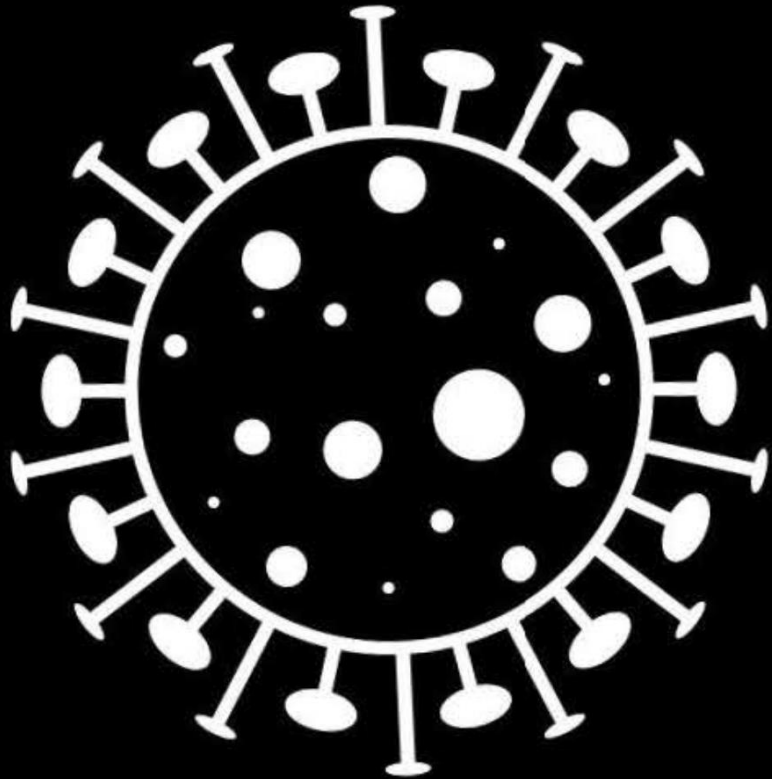
- Decreased white blood cells



TAKING ON THE CHALLENGE

[Grand Challenge](#) [Archives](#) [Reader Studies](#) [Challenges](#) [Algorithms](#)

[Sign In](#) [Register](#)



COVID-19 CT

• *“To mitigate the inefficiency and shortage of existing tests for COVID-19, we propose this competition to encourage the development of effective Deep Learning techniques to diagnose COVID-19 based on CT images.”*

The problem in this challenge is to classify each CT image into positive COVID-19 (the image has clinical findings of COVID-19) or negative COVID-19 (the image does not have clinical findings of COVID-19).

FINDING THE DATA



- Many interesting open data sets available through the Grand Challenge.
- UCSD collected various scans through liaising with local doctors from affected regions in China.
- Not the cleanest, or most comprehensive, data but provided high resolutions for potential deep learning analysis.
- Over 700 images, including limited metadata.

COVID-CT-Dataset: A CT Scan Dataset about COVID-19

Jinyu Zhao
UC San Diego

JIZ077@ENG.UCSD.EDU

Yichen Zhang
UC San Diego

YIZ037@ENG.UCSD.EDU

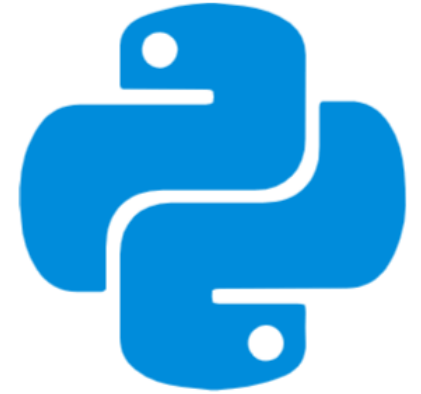
Xuehai He
UC San Diego

X5HE@ENG.UCSD.EDU

Pengtao Xie
UC San Diego, Petuum Inc

PENGTAOXIE2008@GMAIL.COM

PART 1: METADATA ANALYSIS



In hope of uncovering trends and relationships within the patients' medical backgrounds.

METADATA ANALYSIS

	File name	Patient ID	COVID Positive?	Location	Age	Gender	Medical history	Time	Severity	Other diseases	DOI	Captions
0	2020.03.18.20038125-p16-56-6.png	Patient 144	1	"Diamond Princess" Cruise Ship	75.0	M	an increase of extent of GGO with crazy-paving...	On the tenth day of admission	severe	NaN	NaN	['Figure 2. The progress of CT findings in a p...
1	2020.03.18.20038125-p16-56-5.png	Patient 144	1	"Diamond Princess" Cruise Ship	75.0	M	an increase of extent of GGO with crazy-paving...	On the tenth day of admission	severe	NaN	NaN	['Figure 2. The progress of CT findings in a p...
2	2020.03.18.20038125-p16-56-4.png	Patient 144	1	"Diamond Princess" Cruise Ship	75.0	M	an increase of extent of GGO with crazy-paving...	On the tenth day of admission	severe	NaN	NaN	['Figure 2. The progress of CT findings in a p...
3	2020.03.18.20038125-p16-56-3.png	Patient 144	1	"Diamond Princess" Cruise Ship	75.0	M	On the fourth day of admission, he showed tach...	On the day of admission	severe	NaN	NaN	['Figure 2. The progress of CT findings in a p...
4	2020.03.18.20038125-p16-56-2.png	Patient 144	1	"Diamond Princess" Cruise Ship	75.0	M	On the fourth day of admission, he showed tach...	On the day of admission	severe	NaN	NaN	['Figure 2. The progress of CT findings in a p...
...
741	10%0.jpg	crawled_patient_10	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
742	1%2.jpg	crawled_patient_1	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
743	1%1.jpg	crawled_patient_1	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
744	1%0.jpg	crawled_patient_1	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
745	0.jpg	crawled_patient_0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NTM, nontuberculous mycobacterium.

746 rows × 12 columns

- Result of consolidating data from various sources.
- Included metadata of scans, but in need of significant cleaning before analysis.

METADATA ANALYSIS

	Location	Severity	Age	Gender
0	"Diamond Princess" Cruise Ship	Severe	75.0 y/o	Male
1	"Diamond Princess" Cruise Ship	Severe	75.0 y/o	Male
2	"Diamond Princess" Cruise Ship	Severe	75.0 y/o	Male
3	"Diamond Princess" Cruise Ship	Severe	75.0 y/o	Male
4	"Diamond Princess" Cruise Ship	Severe	75.0 y/o	Male
...
344	N/A	Severe	NaN	N/A
345	N/A	Severe	NaN	N/A
346	N/A	Severe	65.0 y/o	Female
347	N/A	Moderate	65.0 y/o	Female
348	N/A	Moderate	65.0 y/o	Female
349 rows × 4 columns				

- Decided on a focused analysis of Location, Severity (of illness), Age, and Gender.

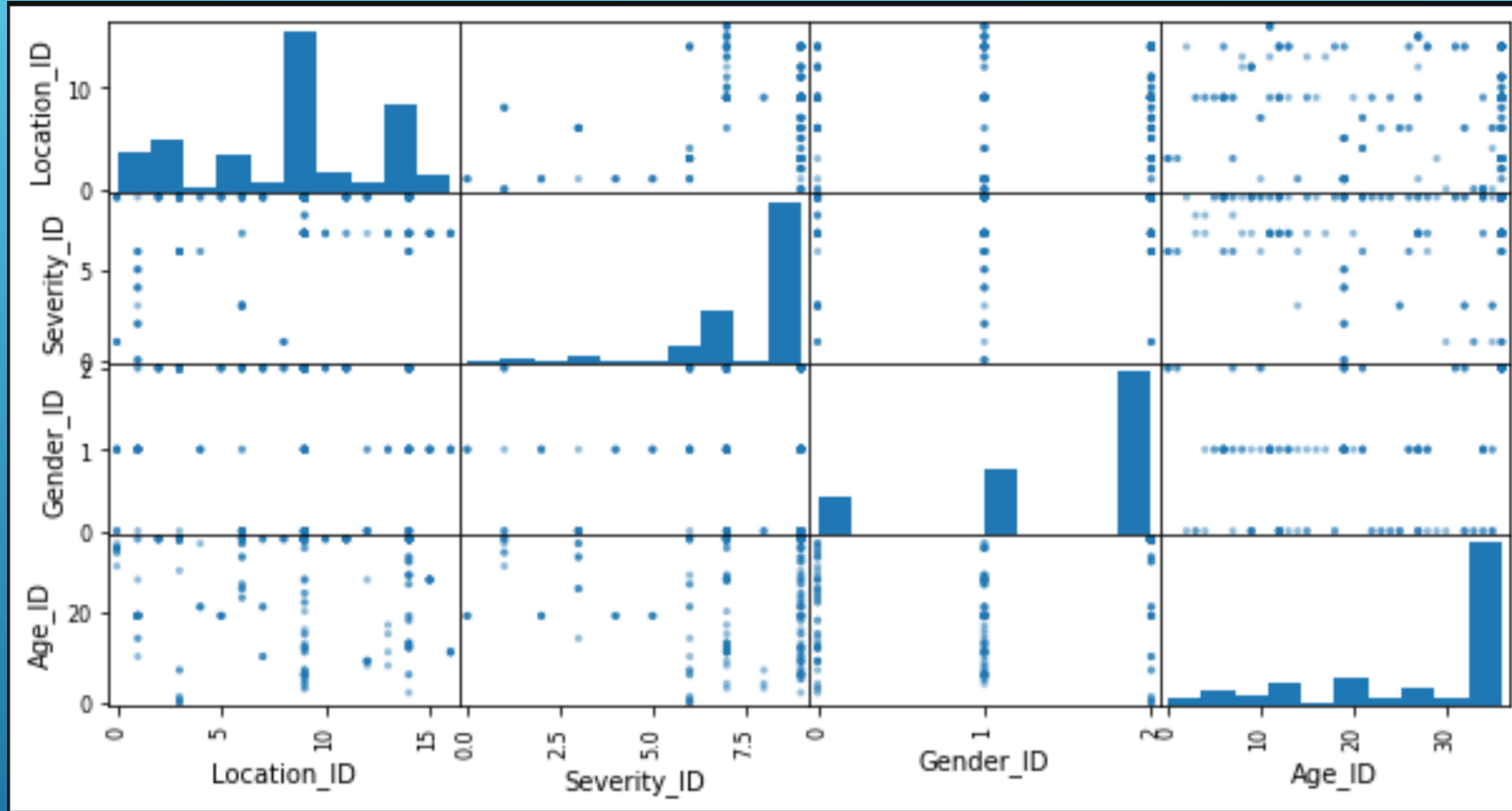
HOT ENCODING NEEDED

- Converted 'Location', 'Severity', 'Gender', and 'Age' string values into *hot encodes* in order to prepare for the sklearn analysis.

	Location_ID	Severity_ID	Gender_ID	Age_ID	Location	Severity	Gender	Age
0	0	9	1	34	"Diamond Princess" Cruise Ship	Severe	Male	75.0 y/o
1	0	9	1	34	"Diamond Princess" Cruise Ship	Severe	Male	75.0 y/o
2	0	9	1	34	"Diamond Princess" Cruise Ship	Severe	Male	75.0 y/o
3	0	9	1	34	"Diamond Princess" Cruise Ship	Severe	Male	75.0 y/o
4	0	9	1	34	"Diamond Princess" Cruise Ship	Severe	Male	75.0 y/o
...
344	9	9	2	36	N/A	Severe	N/A	N/A
345	9	9	2	36	N/A	Severe	N/A	N/A
346	9	9	0	27	N/A	Severe	Female	65.0 y/o
347	9	7	0	27	N/A	Moderate	Female	65.0 y/o
348	9	7	0	27	N/A	Moderate	Female	65.0 y/o

349 rows × 8 columns

NOT SO 'HOT' RESULTS



- Instead of having histograms on the diagonals to display density, we could view the more aesthetically pleasing kernel density estimation(KDE).
- KDE is a non-parametric way to estimate the probability density function of any variable we wish to view.
- No visually discernable trends upon initial analysis.

NOT SO 'HOT' RESULTS

```
## Logistic Regression
```

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

print('Accuracy of logistic regression on training', logreg.score(X_train, y_train))
print('Accuracy of logistic regression on testing', logreg.score(X_test, y_test))
```

```
Accuracy of logistic regression on training 0.41379310344827586
Accuracy of logistic regression on testing 0.3409090909090909
```

```
# K-nearest neighbor
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
print('Accuracy of Knn on training', knn.score(X_train, y_train))
print('Accuracy of Knn on testing', knn.score(X_test, y_test))
```

```
Accuracy of Knn on training 0.45977011494252873
Accuracy of Knn on testing 0.4659090909090909
```

```
## Decision Tree
```

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

print('Accuracy of Decision tree on training', dt.score(X_train, y_train))
print('Accuracy of Decision tree on testing', dt.score(X_test, y_test))
```

```
Accuracy of Decision tree on training 0.5019157088122606
Accuracy of Decision tree on testing 0.5227272727272727
```

```
# Linear Discriminant Analysis
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
print('Accuracy of Lda on training', lda.score(X_train, y_train))
print('Accuracy of Lda on testing', lda.score(X_test, y_test))
```

```
Accuracy of Lda on training 0.4061302681992337
Accuracy of Lda on testing 0.3522727272727273
```

- Above models give a max accuracy of ~50%

NOT SO 'HOT' RESULTS

```
# Gaussian Naive Bayes
```

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Accuracy of GNB on training', gnb.score(X_train, y_train))
print('Accuracy of GNB on testing', gnb.score(X_test, y_test))
```

```
Accuracy of GNB on training 0.17624521072796934
Accuracy of GNB on testing 0.18181818181818182
```

```
# Support Vector Machine
```

```
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
print('Accuracy of SVM on training', svm.score(X_train, y_train))
print('Accuracy of SVM on testing', svm.score(X_test, y_test))
```

```
Accuracy of SVM on training 0.41379310344827586
Accuracy of SVM on testing 0.3409090909090909
```

- Above models give a max accuracy of ~41%
- We can conclude that the metadata itself does not show a robust relationship between the Location, Illness Severity, Age, and Gender of the patients.

PART 2:
CT-SCAN ANALYSIS
USING NEURAL NETWORKS AND MACHINE LEARNING



PyTorch
+
fastai

WHAT ARE CT-SCANS?

(COMPUTED TOMOGRAPHY)



A CT scan, or computed tomography scan, is a medical imaging procedure that uses computer-processed combinations of many X-ray measurements taken from different angles to produce cross-sectional images of specific areas of a scanned object, allowing the user to see inside the object without cutting.

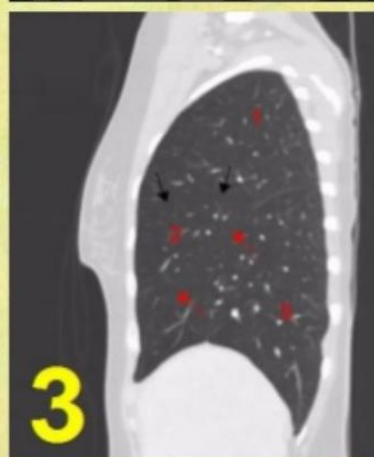
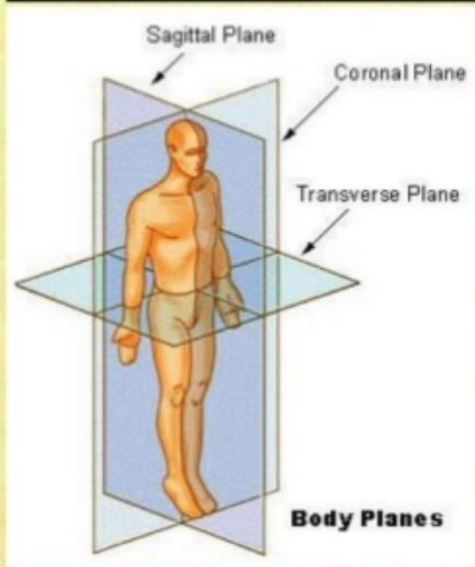
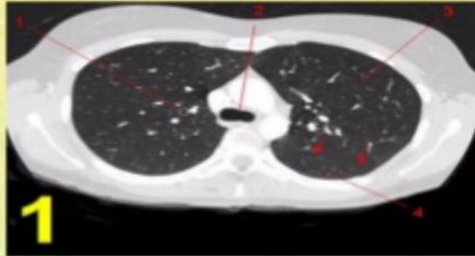
WHAT ARE CT-SCANS?

(COMPUTED TOMOGRAPHY)

CT Chest **Anatomy** – **views**

Three views:

1. **Axial** view
2. **Coronal** view
3. **Sagittal** view



- Our CT scan data set includes over 700 images of *2019-nCoV-Positive* and *2019-nCoV-Negative* patients.
- All scans are from the *transverse* (also known as *axial*) view.

COMPUTATIONAL DATA CRUNCHING



- Our image computational data set will certainly overwhelm most PC hardware.
- Majority of high-end machine learning is performed in the *cloud*, on dedicated server farms and GPU's (graphics processing units).

HORSEPOWER MATTERS

```
[5] print(torch.cuda.current_device())  
    print(torch.cuda.device(0))  
    print(torch.cuda.device_count())  
    print(torch.cuda.get_device_name(0))
```

```
0  
<torch.cuda.device object at 0x7f321b40e240>  
1  
Tesla K80
```

```
[5] print(torch.cuda.current_device())  
    print(torch.cuda.device(0))  
    print(torch.cuda.device_count())  
    print(torch.cuda.get_device_name(0))
```

```
0  
<torch.cuda.device object at 0x7f36b544e2e8>  
1  
Tesla P100-PCIE-16GB
```

- Google Colab allocates specific GPU's based to the data complexity within the notebook.
- Tesla K80: Analysis within *~15 minutes*.
- Tesla P100: Analysis within *~1-2 minutes*.

FIND YOUR “PATH”

```
from google.colab import drive
drive.mount('/content/drive/')

path = Path("/content/drive/My Drive/UCB Bootcamp/Final Project/")
folder='CT_COMBINED'

df = pd.read_csv(path/"covid_all_tags-3.csv")
df
```

	image_name	tags
0	2020.03.18.20038125-p16-56-6.png	2019-nCoV-Positive:Severe:"Diamond Princess" C...
1	2020.03.18.20038125-p16-56-5.png	2019-nCoV-Positive:Severe:"Diamond Princess" C...
2	2020.03.18.20038125-p16-56-4.png	2019-nCoV-Positive:Severe:"Diamond Princess" C...
3	2020.03.18.20038125-p16-56-3.png	2019-nCoV-Positive:Severe:"Diamond Princess" C...
4	2020.03.18.20038125-p16-56-2.png	2019-nCoV-Positive:Severe:"Diamond Princess" C...
...
741	10%0.png	2019-nCoV-Negative
742	1%2.png	2019-nCoV-Negative
743	1%1.png	2019-nCoV-Negative
744	1%0.png	2019-nCoV-Negative
745	0.png	2019-nCoV-Negative


746 rows × 2 columns

- Google Colab also connects seamlessly with Google Drive (naturally).

“MODEL” BEHAVIOR

```
[16] arch = models.resnet34  
      arch = models.resnet50  
      arch = models.densenet121
```

```
[17] acc_02 = partial(accuracy_thresh, thresh=0.2)  
      f_score = partial(fbeta, thresh=0.2)  
      learn = cnn_learner(data, arch, metrics=[acc_02, f_score])
```

```
☐→ Downloading: "https://download.pytorch.org/models/densenet121-a639ec97.pth" to /root/.cache/torch/checkpoints/densenet121-a639ec97.pth  
100%  30.8M/30.8M [00:00<00:00, 61.2MB/s]
```

- For the purposes of this project, we will be using the *Resnet34*, *Resnet50*, and *Densenet121* models.

LOADING THE IMAGES

```
[ ] np.random.seed(42) # set random seed so we always get the same validation set
src = (Imagelist.from_csv(path, 'covid_all_tags-3.csv', folder='CT_COMBINED', suffix='')
      # Load data from csv
      .split_by_rand_pct(0.2)
      # split data into training and validation set (20% validation)
      .label_from_df(label_delim=':')
      # label data using the tags column (second column is default)
      )
```

```
[ ] data = (src.transform(size=128) #
            .databunch().normalize(imagenet_stats))
```

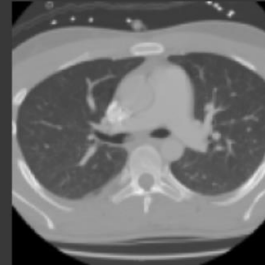
```
[ ] data.show_batch(rows=3, figsize=(12,9))
```



2019-nCoV-Negative



2019-nCoV-Negative



2019-nCoV-Negative



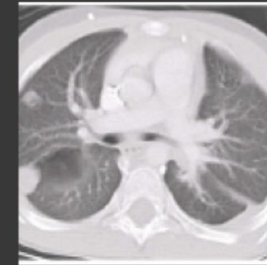
2019-nCoV-Positive:China (Unspecified Region);Severe



2019-nCoV-Positive;Severe



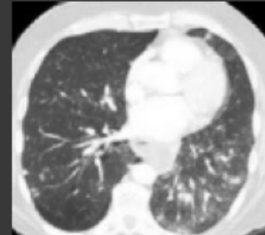
2019-nCoV-Negative



2019-nCoV-Negative



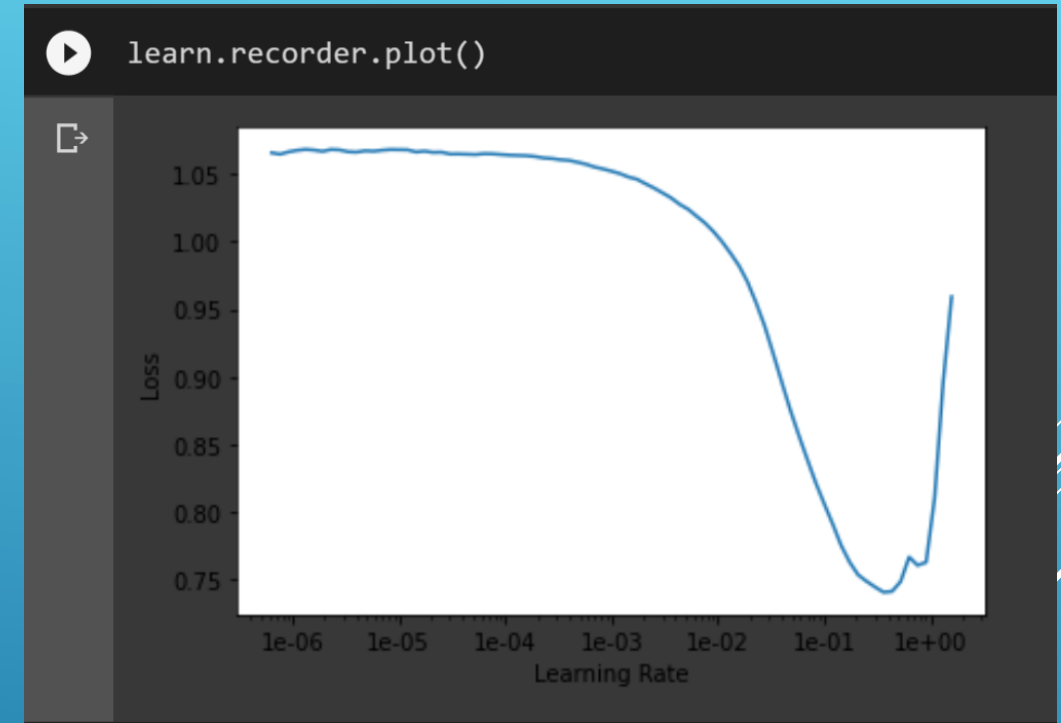
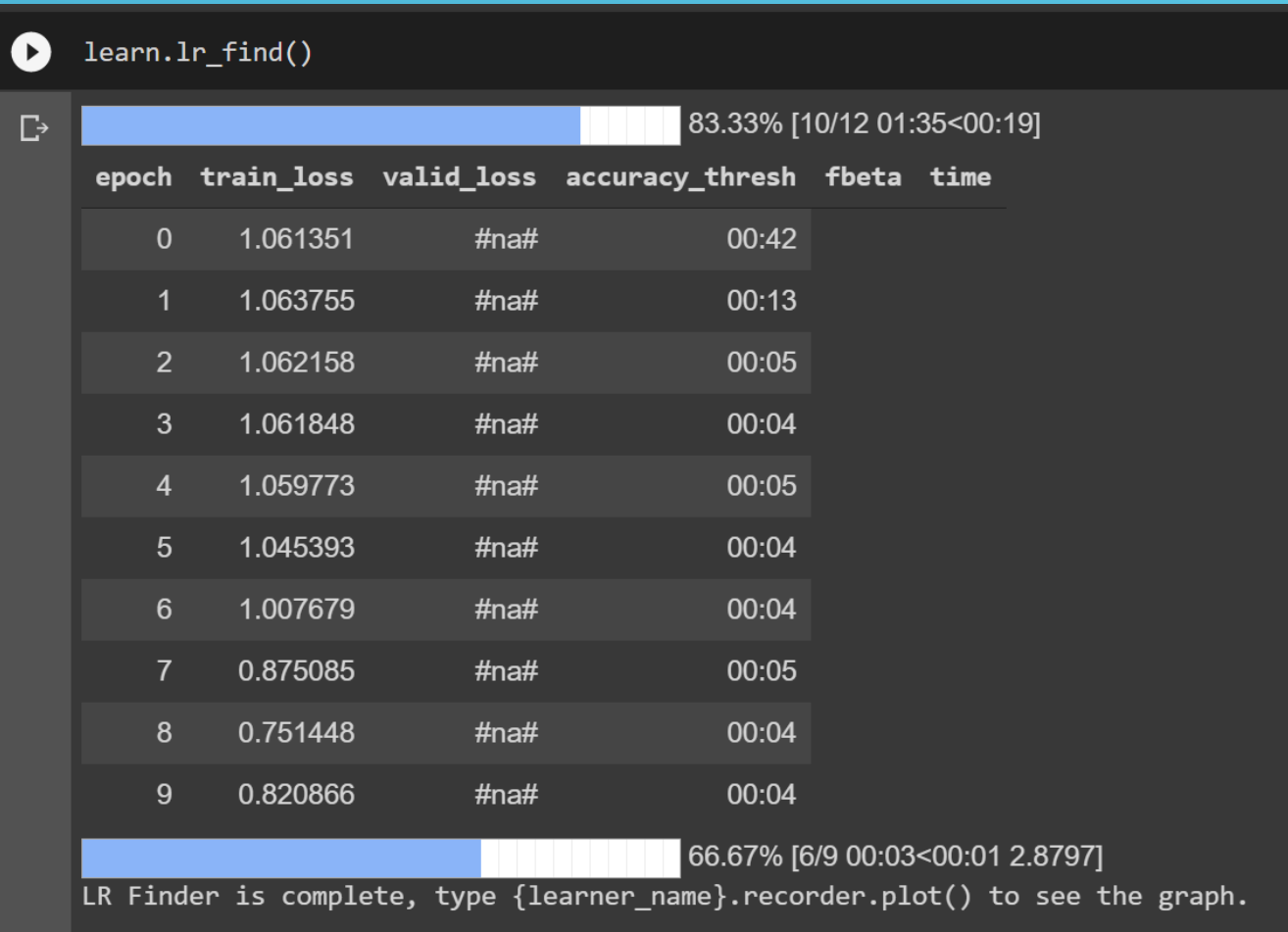
2019-nCoV-Negative



2019-nCoV-Negative



FINDING THE OPTIMAL LR



- Finding the optimum learning rate is key to training an accurate model.
- A suitable learning rate (lr) is 1e-01 here (0.01).

TRAINING THE MODEL

```
[20] lr = 0.01
```

```
[21] learn.fit_one_cycle(10, lr)
```

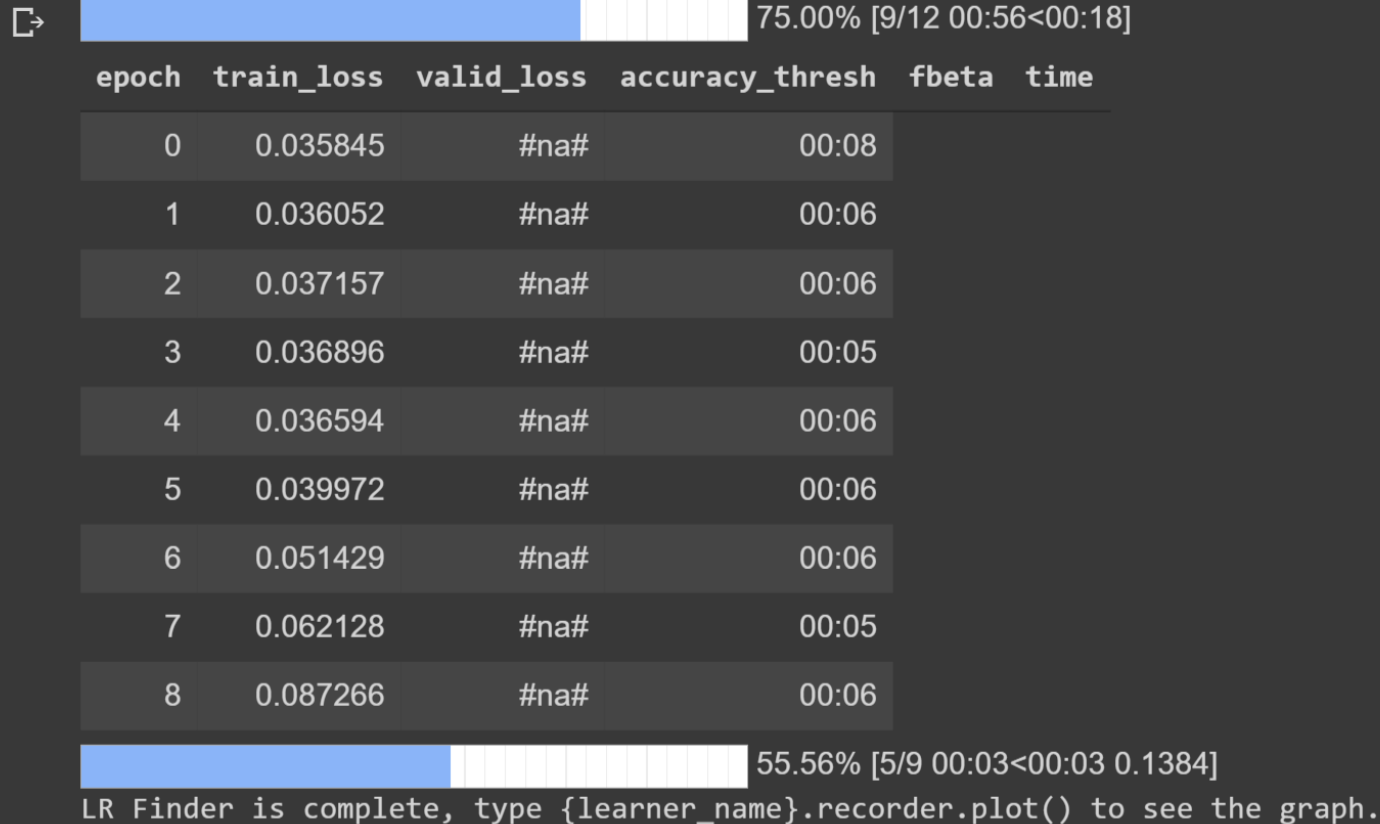
epoch	train_loss	valid_loss	accuracy_thresh	fbeta	time
0	1.032855	0.940297	0.152329	0.127207	00:24
1	0.958101	0.708277	0.191682	0.124609	00:06
2	0.808644	0.282720	0.585926	0.191954	00:06
3	0.605371	0.102352	0.971019	0.599506	00:06
4	0.459624	0.091033	0.974781	0.682543	00:06
5	0.359051	0.077684	0.977324	0.727621	00:06
6	0.287515	0.072184	0.979459	0.756329	00:06
7	0.234119	0.072399	0.979967	0.748680	00:06
8	0.194112	0.070377	0.980476	0.757053	00:06
9	0.162802	0.068787	0.980069	0.758756	00:06

- Using our determined learning rate, we begin to train our model with the CT scan data set, in this case setting 10 epochs.
- fbeta scores are the calculated mean of the precision and recall rate.

TIME TO FINE TUNE OUR MODEL..

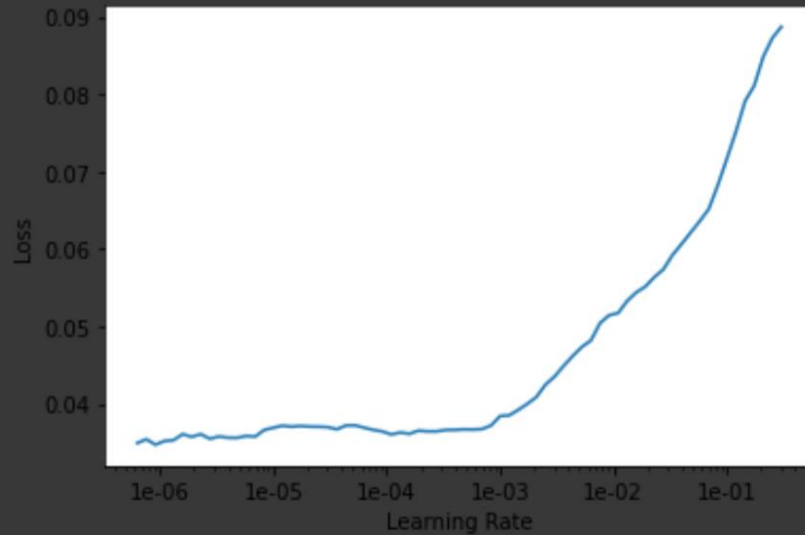
```
[23] learn.unfreeze()
```

```
[24] learn.lr_find()  
learn.recorder.plot()
```



- We can 'unfreeze' the layers of our image files, enabling our learner to scan more than just the superficial top layer of each image.

TIME TO FINE TUNE OUR MODEL..



```
[25] # fit model with differential learning rates  
learn.fit_one_cycle(5, slice(1e-4, lr/5))
```

epoch	train_loss	valid_loss	accuracy_thresh	fbeta	time
0	0.038440	0.061754	0.980273	0.779437	00:07
1	0.037127	0.077181	0.978341	0.704675	00:07
2	0.038406	0.080687	0.981391	0.761305	00:07
3	0.036424	0.063935	0.982713	0.779770	00:07
4	0.033513	0.059823	0.982408	0.780589	00:07

- Unfreezing the model gives us a new learning rate for the deeper scan.
- We now retrain our model again with the new optimal learning rate (1e-4).

INCREASE RESOLUTION TO FURTHER FINE TUNE OUR MODEL AND PREVENT OVERFITTING

```
[34] data = (src.transform(size=256)
            .databunch(bs=16).normalize(imagenet_stats))
learn.data = data
```

```
[35] learn.freeze() # freeze bottom layers again
```

```
[36] learn.lr_find()
learn.recorder.plot()
```

66.67% [2/3 00:27<00:13]

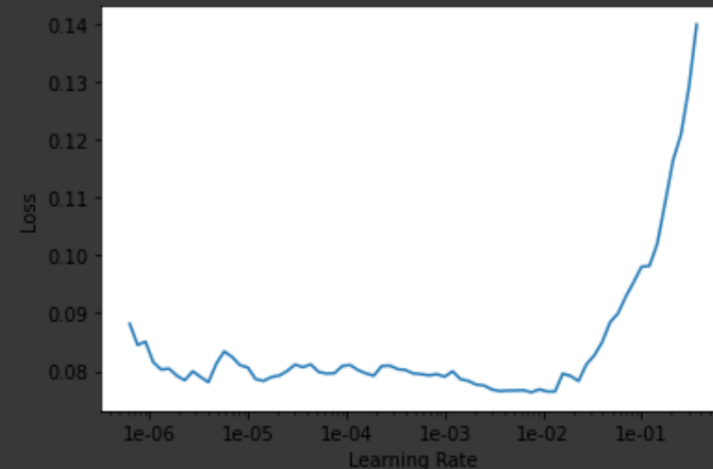
epoch	train_loss	valid_loss	accuracy_thresh	fbeta	time
-------	------------	------------	-----------------	-------	------

0	0.079636	#na#	00:13		
---	----------	------	-------	--	--

1	0.092887	#na#	00:13		
---	----------	------	-------	--	--

35.14% [13/37 00:05<00:09 0.2803]

LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

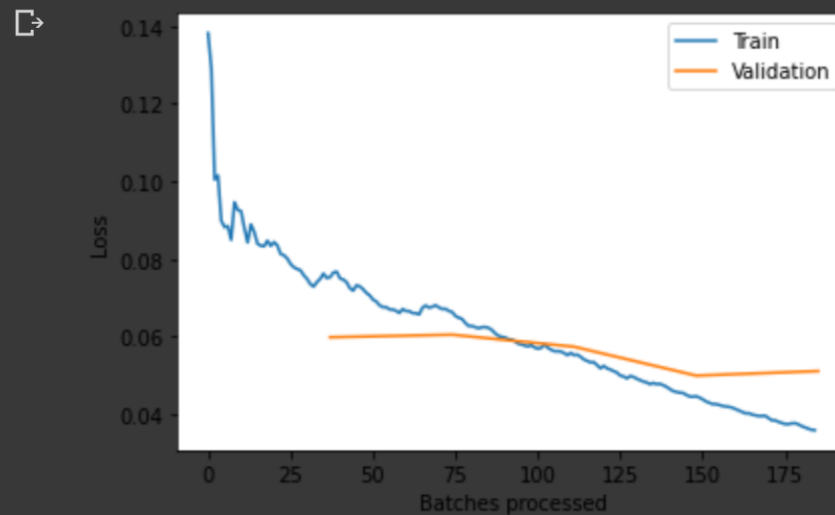


INCREASE RESOLUTION TO FURTHER FINE TUNE OUR MODEL AND PREVENT OVERFITTING

```
[37] lr=1e-2/2  
learn.fit_one_cycle(5, slice(lr))
```

epoch	train_loss	valid_loss	accuracy_thresh	fbeta	time
0	0.075115	0.059854	0.979662	0.787462	00:18
1	0.066634	0.060508	0.976815	0.777560	00:15
2	0.055821	0.057406	0.982306	0.809753	00:15
3	0.044508	0.049963	0.984340	0.827394	00:15
4	0.035905	0.051125	0.983527	0.818785	00:15

```
[38] learn.recorder.plot_losses()
```



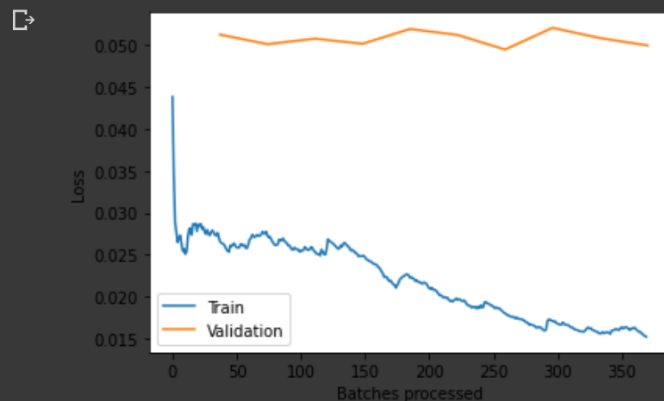
INCREASE RESOLUTION TO FURTHER FINE TUNE OUR MODEL AND PREVENT OVERFITTING

```
[39] learn.unfreeze()
```

```
[40] learn.fit_one_cycle(10, slice(1e-6, 1r/10))
```

epoch	train_loss	valid_loss	accuracy_thresh	fbeta	time
0	0.027091	0.051225	0.984442	0.830691	00:22
1	0.027763	0.050092	0.984849	0.828466	00:20
2	0.025481	0.050744	0.984543	0.813441	00:20
3	0.024824	0.050138	0.983628	0.815159	00:20
4	0.022535	0.051897	0.985255	0.823181	00:20
5	0.019802	0.051186	0.985560	0.828295	00:20
6	0.018137	0.049453	0.984849	0.818405	00:20
7	0.017159	0.052033	0.985560	0.827723	00:20
8	0.015658	0.050815	0.985459	0.829405	00:20
9	0.015235	0.049940	0.985764	0.832600	00:20

```
[41] learn.recorder.plot_losses()
```



- Final accuracy threshold at 98.58% (!)
- Validation losses have leveled off, indicating that our model has learned and matured in regards to our training.

PRETRAINING MODEL COMPARISONS

epoch	train_loss	valid_loss	accuracy_thresh	fbeta	time
0	0.040898	0.060281	0.980951	0.788169	00:10
1	0.037637	0.061887	0.980754	0.785149	00:10
2	0.035388	0.060782	0.980162	0.777024	00:10
3	0.031691	0.059615	0.980754	0.773378	00:10
4	0.029712	0.060743	0.980063	0.764202	00:10
5	0.026289	0.058449	0.982531	0.784286	00:10
6	0.023814	0.058649	0.981840	0.788380	00:10
7	0.020677	0.059324	0.982136	0.784036	00:10
8	0.019066	0.058761	0.982827	0.793286	00:10
9	0.018699	0.058851	0.982629	0.782082	00:10

• Resnet34

epoch	train_loss	valid_loss	accuracy_thresh	fbeta	time
0	0.034337	0.052087	0.982728	0.824413	00:21
1	0.033525	0.054058	0.983518	0.815720	00:19
2	0.031726	0.054804	0.984110	0.823852	00:19
3	0.028342	0.052801	0.984702	0.825476	00:19
4	0.026116	0.051547	0.985491	0.828432	00:19
5	0.024015	0.052921	0.984801	0.822138	00:19
6	0.021106	0.052064	0.985689	0.834777	00:19
7	0.018530	0.051617	0.985886	0.828782	00:19
8	0.016385	0.049866	0.986478	0.835613	00:19
9	0.015673	0.050505	0.985886	0.831158	00:19

• Resnet50

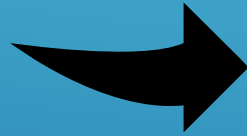
epoch	train_loss	valid_loss	accuracy_thresh	fbeta	time
0	0.027091	0.051225	0.984442	0.830691	00:22
1	0.027763	0.050092	0.984849	0.828466	00:20
2	0.025481	0.050744	0.984543	0.813441	00:20
3	0.024824	0.050138	0.983628	0.815159	00:20
4	0.022535	0.051897	0.985255	0.823181	00:20
5	0.019802	0.051186	0.985560	0.828295	00:20
6	0.018137	0.049453	0.984849	0.818405	00:20
7	0.017159	0.052033	0.985560	0.827723	00:20
8	0.015658	0.050815	0.985459	0.829405	00:20
9	0.015235	0.049940	0.985764	0.832600	00:20

• Densenet121

CONFUSED?

HIGH LEVEL OVERVIEW:

- Load images
- Find learning rate of model
- Train model based on learning rate
- Unfreeze images to scan deeper
- Find unfrozen learning rate
- Train model based on new, unfrozen learning rate



- Increase image resolution and reload images
- Find new learning rate of model
- Train model based on new learning rate
- Unfreeze images to scan deeper
- Find new unfrozen learning rate
- Retrain model based on new, unfrozen learning rate

GOT THE SWEET .PKL!
TIME TO DEPLOY!

.pkl file

Contains all the weights,
measures, and calculations
of our machine learning
training.

<https://covid19-ct-ai.onrender.com/>

ACKNOWLEDGEMENTS

- D.R.E.W. 🏰
- Jeremy Howard (fast.ai) 🤖
- UCSD School of Medicine 🏥
- Healthcare workers in China during the initial outbreak (and beyond) 🏠



QUESTIONS? COMMENTS?

