Project 4

Using Image Classification to
Detect the Presence of COVID-19 in Lung X-Rays

Group 3: Joshua Peterson, Michael Gray and Mangala Desai

Overview

- Introduction & Motivation for Project
- Background Research
- Data Description
- Methods
  - Data Cleaning & Preparation
  - Model Building
  - Evaluation
- Results
  - Analysis of Models
  - Selected Models
- Discussion & Conclusions

Introduction & Motivation for Project

- The need to develop methods for accurately and quickly identifying viral pneumonias related to COVID-19 is a current and urgent need.
- Deep learning models are being deployed currently to detect the effects of COVID-19 within radiology images; however, it is still a developing field.
  - These models can help improve accuracy of radiologists and even replace one in the event of a shortage.
- We aim to create a deep learning model that accurately detects COVID-19 and other types of pneumonia that is not biased by common artifacts such as feeding tubes and image quality differences. Many of the existing models are trained on x-ray images of lungs AND surrounding areas.

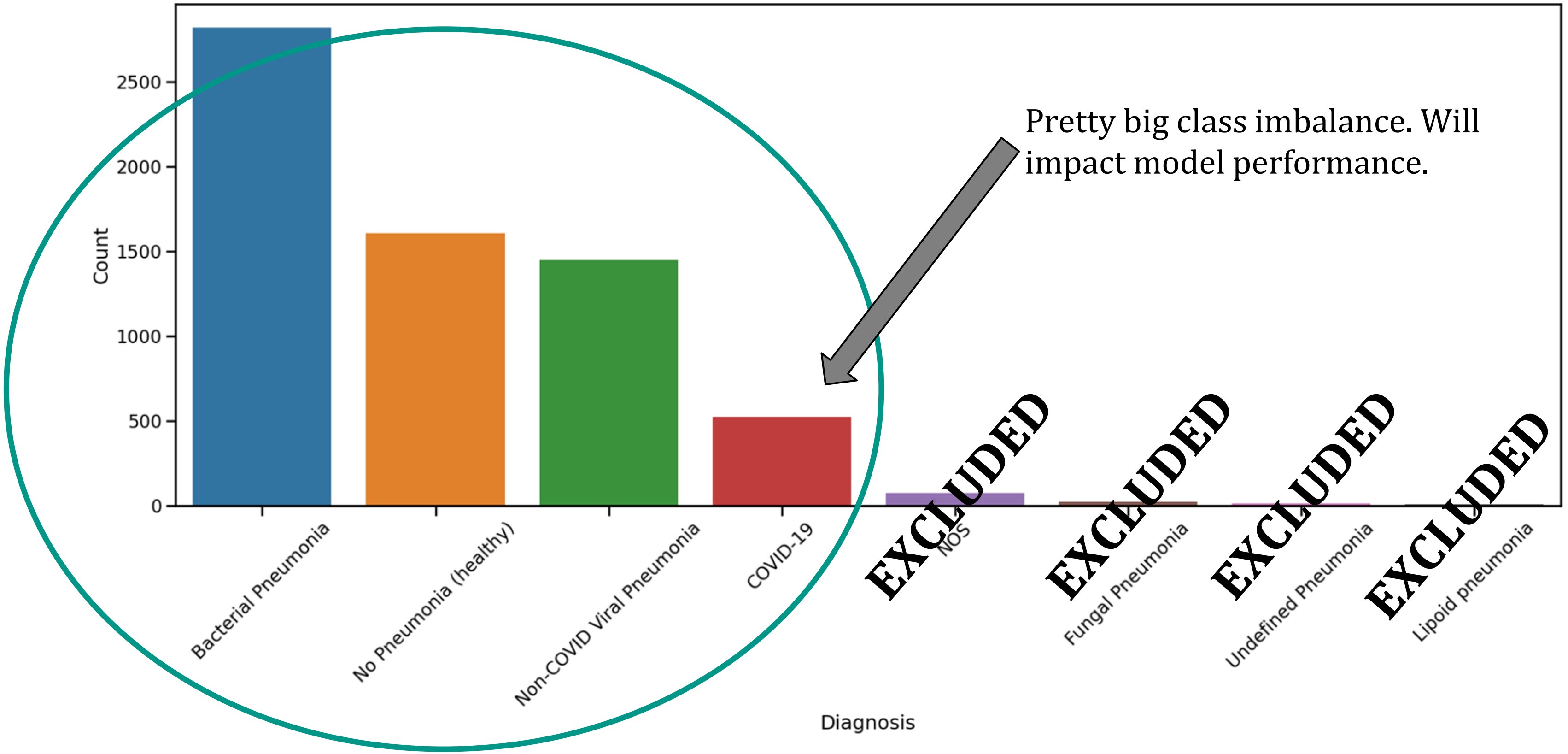
Background Research

- The utilization of deep learning techniques to detect COVID-19 is a developing field
  - Various deep learning architectures have been suggested by other researchers to detect COVID-19 in radiological images, such as VGG19, ResNet50, Xception and InceptionV3.
  - Lack of consensus in the field
- There have been issues with models being able to generalize when deployed
  - To be able to use such models practically, they would need to be able to generalize to new data well.
- Researchers trained a 121-layer convolutional neural network on a dataset of 100,000 lung x-ray images that were labeled for 14 different diseases. Unsure how transferable this network is when detecting COVID-19.

Data Description

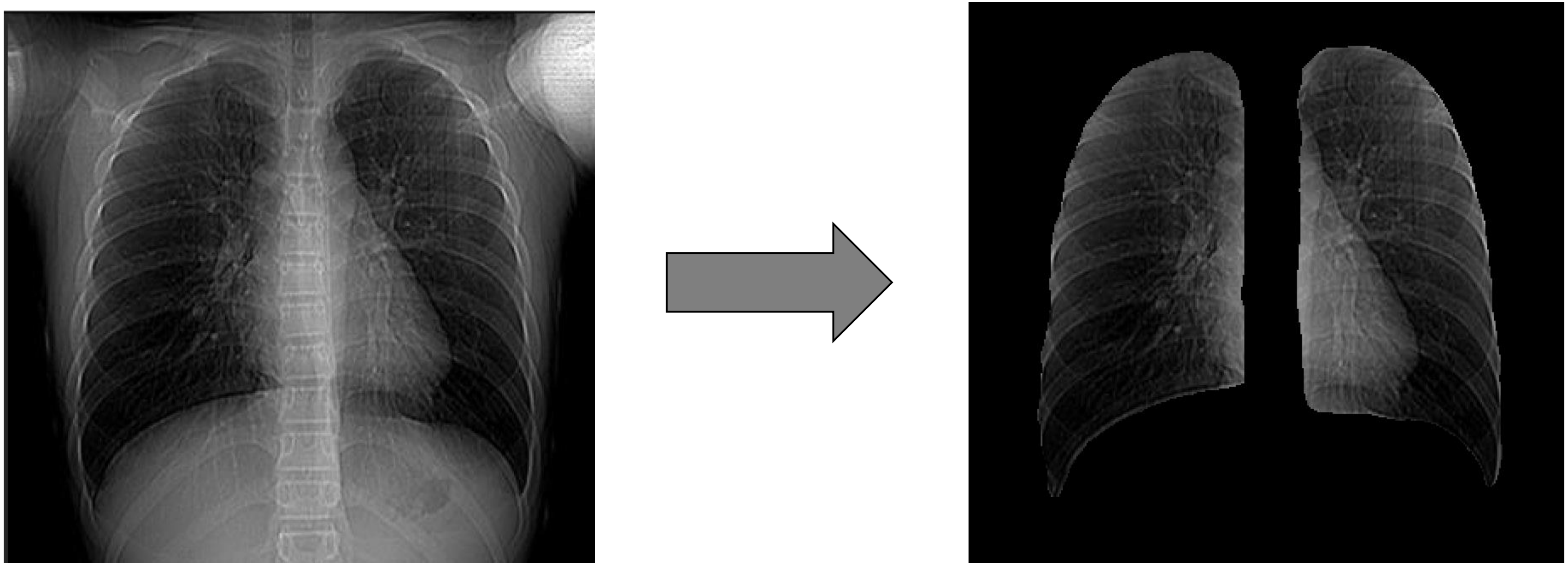
- The dataset contains 6,504 JPEG and PNG images. There are corresponding JSON files for each image that contain the annotations and metadata for the images.
- The classes of interest among the images are the following: viral pneumonia (non-COVID), bacterial pneumonia, no pneumonia (healthy) and COVID-19.
- All cases of COVID-19 also contain viral pneumonia.
- Fungal and Lipoid Pneumonia labels comprise a small percentage. 74 were not otherwise label.
- These low N classes will be excluded.

Data Description



Pretty big class imbalance. Will impact model performance.

Methods - Data Cleaning & Preparation



Segmentation coordinates applied to raw images. Masking should help generalizability by reducing external cues that bias the model.

Methods - Data Cleaning & Preparation

Folder Structure

```
├── masked_images/
│    ├── train/
│    │    ├── healthy/
│    │    ├── bacterial/
│    │    ├── viral/
│    │    └── covid/
│    │
│    └── test/
│         ├── healthy/
│         ├── bacterial/
│         ├── viral/
│         └── covid/
```

- **70/30 train/test split.**
- **Train split 75/25 for validation data**

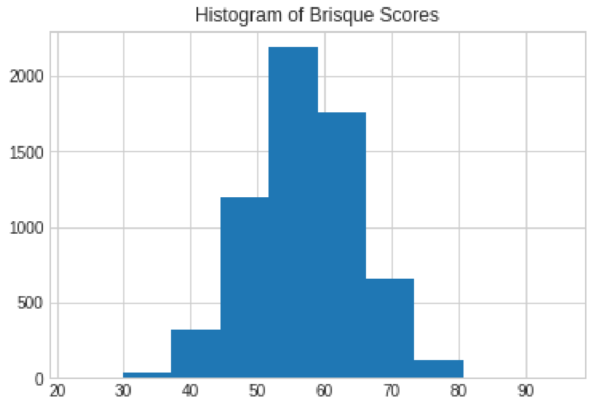
(same structure also for raw_images)

Methods - Data Cleaning & Preparation

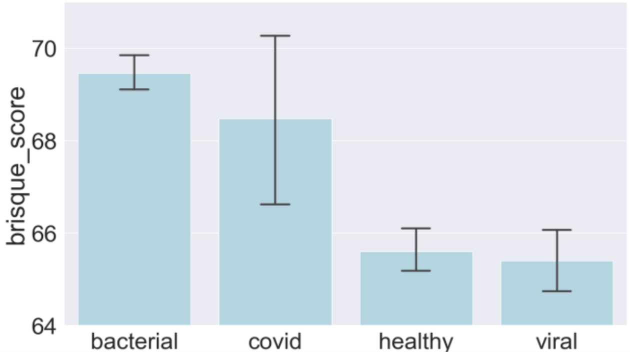
Determining the Quality of Our Images

- We calculated a BRISQUE score for each of our images to assess the quality of the images.

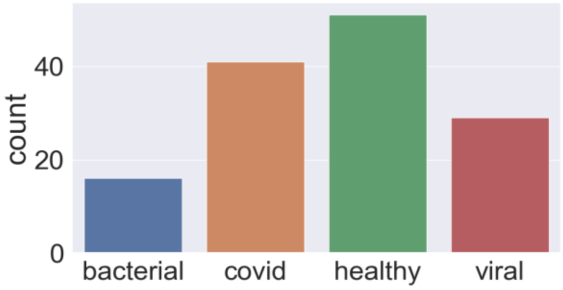
Overall:



Score vs class:



1.5*IQR Outliers:



Significantly worse image quality for covid and bacterial compared to healthy ($p<0.05$)

Later we will compare removing outliers vs. not to see if it influences model accuracy.

Methods - Initial Modeling

Initially developed CNN from scratch

- Used as a starting point from which to further develop and refine a model for this image classification task.

Initial 16 Layer CNN Model

```
class_weight = {0: 1.00,
                1: 1.75,
                2: 1.94,
                3: 6.40}

model = tf.keras.Sequential([
    layers.Rescaling(1./255),
    layers.RandomRotation(0.1, fill_mode='constant',fill_value=0),
    layers.RandomZoom(0.1),
    layers.RandomTranslation(0.1,0.1,fill_mode='constant',fill_value=0),
    layers.Conv2D(64, (3,3), activation = 'relu', input_shape = (200,200,3)),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(128, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(128, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(256, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(256, (3,3), activation = 'relu'),
    layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(512, activation=tf.nn.relu),
    layers.Dropout(0.2),
    layers.Dense(nb_classes, activation=tf.nn.softmax)
])

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

Methods - Results of Initial Modeling

Making Predictions with this Initial Model

- Able to achieve a test accuracy of 75%
- The classification matrix shows how well the model performs with regard to each class
    - Best at identifying healthy lungs
    - Struggles with identifying viral pneumonia the most.
    - Still does not get close to state-of-the-art models in the field.

57/57 [==============================] - 2s 26ms/step

	precision	recall	f1-score	support
0	0.84	0.67	0.75	771
1	0.89	0.89	0.89	477
2	0.53	0.69	0.60	421
3	0.75	0.90	0.82	124
accuracy			0.75	1793
macro avg	0.75	0.79	0.76	1793
weighted avg	0.77	0.75	0.76	1793

```
# Evaluating the model on the test dataset
test_model = keras.models.load_model('img_class_model.keras')
test_loss, test_acc = test_model.evaluate(test_images, test_labels)
print(f"Test accuracy: {test_acc:.3f}")
```

```
57/57 [==============================] - 2s 28ms/step - loss: 0.5832 - accuracy: 0.7513
Test accuracy: 0.751
```

Methods - Testing Pretrained Models

- These pre-trained model were run without class weighting.
- The following pre-trained models were tested:
    - VGG16
    - VGG19
    - ResNet50
    - Xception
    - InceptionV3
- These models were selected to be tested based on what we've seen used in other research in the field.

```
inputs = keras.Input(shape=(200, 200, 3))
x = layers.Rescaling(1./255),
x = data_augmentation(inputs)
x = keras.applications.resnet50.preprocess_input(x)
x = class_model_base(x) # using the ResNet pre-trained model as the base for my
x = layers.Flatten()(x)
x = layers.Dense(512, activation=tf.nn.relu)(x)
x = layers.Dropout(0.2)(x) # Used a dropout rate of 0.2 to regularize the model
outputs = layers.Dense(nb_classes, activation=tf.nn.softmax)(x)
model = keras.Model(inputs, outputs)

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

Results - Testing Pretrained Models

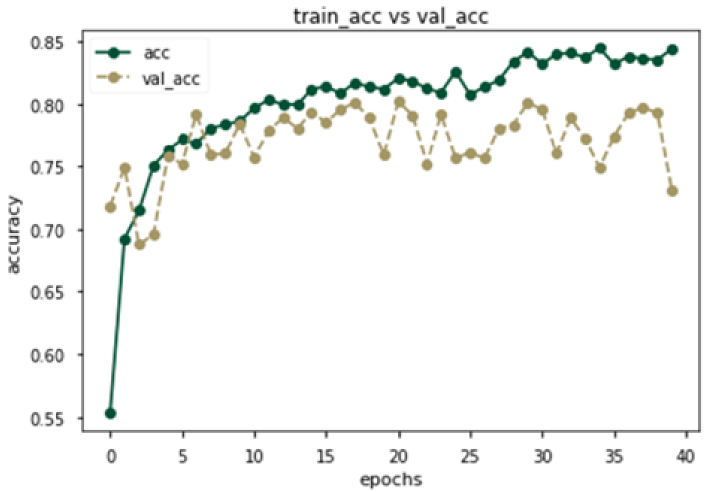
	VGG16	VGG19	ResNet50	Xception	InceptionV3
Test Accuracy	0.816	0.799	0.833	0.785	0.76
Macro Avg. Precision	0.82	0.8	0.84	0.76	0.75
Macro Avg. Recall	0.84	0.82	0.84	0.82	0.78
Bacterial F1 Score	0.83	0.8	0.84	0.8	0.79
Healthy F1 Score	0.91	0.92	0.92	0.89	0.87
Viral F1 Score	0.66	0.64	0.69	0.62	0.51
COVID-19 F1 Score	0.91	0.88	0.91	0.84	0.84

- The ResNet50 pre-trained model performed the best with a test accuracy of 83.3%

Results - Testing Pretrained Models

- Because ResNet50 performed the best, additional fine tuning was conducted.
- Unfroze the last 32 layers of the ResNet50 model. The entirety of the “conv5” block.
- Brought the accuracy of the model up to **85.7%**.

	ResNet50	Xception
Test Accuracy	0.857	0.831
Macro Avg. Precision	0.87	0.82
Macro Avg. Recall	0.87	0.83
Bacterial F1 Score	0.86	0.84
Healthy F1 Score	0.95	0.92
Viral F1 Score	0.73	0.7
COVID-19 F1 Score	0.95	0.83



Results - Testing Pretrained Models

- Chose to use the fine-tuned ResNet50 pre-trained model.
- We were able to increase our test accuracy by 10% by using this model.
- Achieved a final accuracy of 86% with this model.
- However, we still wanted to explore PyTorch implementation of our CNN model

```
100/100 [==============================] - 9s 82ms/step
              precision    recall  f1-score   support

           0       0.86      0.87      0.86      1405
           1       0.97      0.92      0.95       816
           2       0.72      0.73      0.73       734
           3       0.91      0.95      0.93       223

    accuracy                           0.86      3178
   macro avg       0.87      0.87      0.87      3178
weighted avg       0.86      0.86      0.86      3178
```

Methods - Migrating to PyTorch

Why?

- More model control
- More room for optimization
- Better Apple M1 chip support (for GPU acceleration)
- Faster than Keras

How?

- Use conda to create a virtual environment with python 3.9.
- conda install pytorch torchvision torchaudio -c pytorch
- In the model script:

```
# mps is for the M1 chip on MacBook Pro
device = torch.device("mps")
```

```
model = CNN16Model()
model.to(device)
```

```
loss_fn = nn.CrossEntropyLoss()
loss_fn.to(device)
```

```
inputs = inputs.to(device)
labels = labels.to(device)
```

- Then, ask ChatGPT how to convert your Keras model to PyTorch! Just kidding ... Sort of ...

Methods - Migrating to PyTorch

CNN Architecture:

```
class CNN16Model(nn.Module):
    def __init__(self):
        super().__init__()
        self.network = nn.Sequential(
            nn.Conv2d(1, 64, 3),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(64, 64, 3),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(64, 128, 3),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(128, 128, 3),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(128, 256, 3),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Conv2d(256, 256, 3),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.Flatten(),
            nn.Linear(256, 512),
            nn.Dropout(0.2),
            nn.Linear(512, 4)
        )

    def forward(self, x):
        return self.network(x)
```

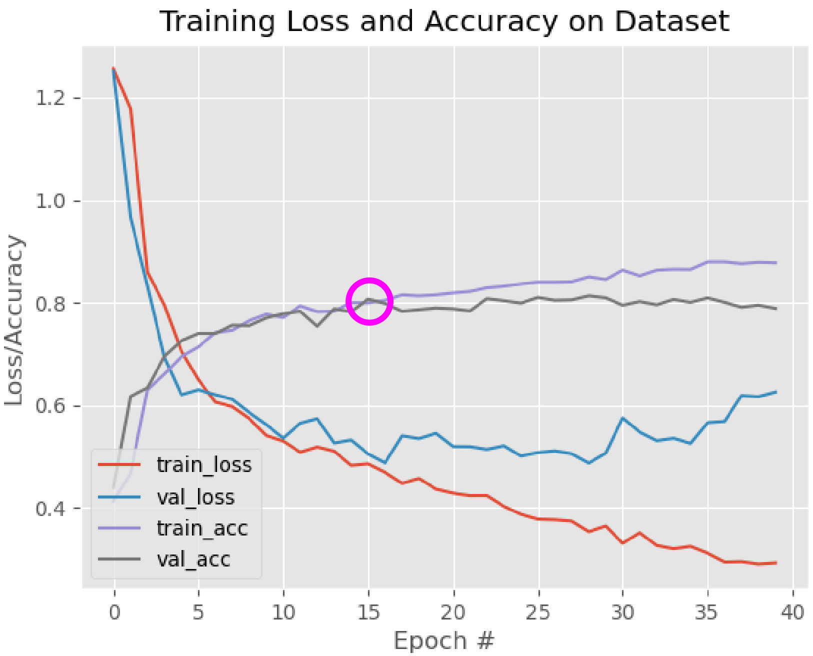
- 6 Conv2d layers
- 1 input channel in 1st layer (grayscale images)
- 3x3 filter kernel
- 2x2 MaxPool2d
- ReLU activations
- 2 fully connected layers

Training was noticeably faster after enabling M1 GPU, compared to both CPU and free-tier Colab.

Results - Model Iterations Phase Two

First, plotting the loss and accuracy metrics for the train and validation datasets to find the optimal epoch #

Chose 15 epochs for all final iterations.



Results - Model Iterations Phase Two

Systematically testing the contribution of:

class weights, image normalization, image augmentation, BRISQUE outliers, masked vs raw images.

Run	Dataset	Class Weights?	Image Normalization?	Image Augmentation?	BRISQUE Outliers Removed?	Test Accuracy	Other Notes
1	Masked	No	Yes	Yes	No	81%	Used 40 epochs to tune ideal epoch #. Sweet spot is 15 epochs.
2	Masked	No	Yes	Yes	No	80.50%	15 epochs for this and all other runs. **This is the benchmark for the rest.**
3	Masked	Yes	Yes	Yes	No	74.60%	**Class weights reduced accuracy by 6%**
4	Raw	Yes	Incorrect values	Yes	No	76.20%	Accuracy only improved 1.6% here with the full raw images.
5	Raw	No	Incorrect values	Yes	No	77.70%	Improvement again by removing class weights
6	Raw	No	Yes (correct values)	Yes	No	79.80%	Can compare this to Run 2. **No improvement from using full raw images.**
7	Masked	No	No	Yes	No	79.60%	**Removing image normalization reduced accuracy by 0.9%**
8	Masked	No	Yes	No	No	77.00%	**Model started overfitting at epoch 8. Test acc 3.5% worse than benchmark**
9	Masked	No	Yes	Yes	Yes	79.40%	**Possible reduction in accuracy after removing outliers.**
10	Masked	No	Yes	Yes	No	81.50%	**1% variability in run-to-run performance (compare to run #2)**

<u>Surprise</u>: Class weights reduced accuracy by 6%
<u>Not surprise</u>: Removing image augmentation caused early overfitting and reduced accuracy.
<u>Not ideal</u>: ~1% variability in test-retest accuracy.

Results - Improving Performance of CNN Model

Previously....

- In Keras, we used the same CNN architecture and obtained 75% accuracy.

Now:

- With our best CNN model, we obtained **81.5% accuracy.**

```
Test Loss = 0.5206347907844343 Test Accuracy = 0.7944535073409462
              precision    recall  f1-score   support

   bacterial       0.77      0.89      0.83       555
       covid       0.83      0.71      0.76        82
     healthy       0.83      0.93      0.88       305
       viral       0.79      0.48      0.60       284

    accuracy                           0.79      1226
   macro avg       0.80      0.75      0.77      1226
weighted avg       0.79      0.79      0.78      1226
```

- Did not use class weights in loss function
- Had **image normalization** (by mean and sd)
- Had image augmentation (small random rotations and translations)
- Used masked images
- Our previous best in Keras did not use image normalization.

Discussion & Conclusions

Discussion

- We tried to reach the 95% accuracy of a reference paper using the same base architecture as them.
- We speculate that we were disadvantaged by class imbalance.
  - They had n = 2600 for each of 3 classes.
  - We also had 4 classes.
- There's probably additional image augmentations and preprocessing we could explore to improve the results of our modeling.
- A future study quantifying the effect of class imbalance would be helpful.

Conclusions

- Our model is not sufficient to be useful in a production environment, but we have laid a good foundation for future improvements and efforts.
- Were able to achieve a test accuracy of **85.7%** using a fine tuned ResNet50 pre-trained model
- Developed a PyTorch implementation to improve accuracy of CNN model to **81.5%**

References

1. Fauci AS, Lane HC, Redfield RR. Covid-19 - Navigating the Uncharted. The New England Journal of Medicine [Internet]. 2020 Mar 26 [cited 2022 Sep 13]. Available from: https://www.nejm.org/doi/full/10.1056/nejme2002387
2. Coronavirus World Map: Tracking the Global Outbreak. New York Times. 2022 [cited 2022 Sep 12]. Available from: https://www.nytimes.com/interactive/2021/world/covid-cases.html
3. Del Rio C, Collins LF, Malani P. Long-term Health Consequences of COVID-19. JAMA [Internet]. 2020 Oct 5 [cited 2022 Sep 13]. Available from: https://jamanetwork.com/journals/jama/article-abstract/2771581
4. Sharma O, Sultan AA., Ding H, Triggle CR. A Review of the Progress and Challenges of Developing a Vaccine for COVID-19. Frontiers in Immunology [Internet]. 2020 [cited 2022 Sep 15]; 11. Available from: https://www.frontiersin.org/articles/10.3389/fimmu.2020.585354/full
5. Khan AI, Shah JL, Bhat MM. CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images. Computer Methods and Programs in Biomedicine [Internet]. Available from: https://www.sciencedirect.com/science/article/pii/S0169260720314140?casa_token=YR3Bry_IGi8AAAAA:yzWkADLKpzyF0-R-lCckM7rjKT0rL2i5oibgOImOqGyq7dTf9JnYT-whcjsv6K4fQ920pmfW16c
6. Narin, A., Kaya, C. & Pamuk, Z. Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks. *Pattern Anal Applic* 24, 1207–1220 (2021). 2021 May 9 [cited 2022 Sep 13]. Available from: https://doi.org/10.1007/s10044-021-00984-y
7. Arbib MA. The Handbook of Brain Theory and Neural Networks. MIT Press. 2002 [cited 2022 Sep 15]. Available from: https://dl.acm.org/doi/abs/10.5555/572531
8. Hebb DO. The Organization of Behavior: A Neuropsychological Theory. 99th ed. The United States of America: John Wiley & Sons; 1949 [cited 2022 Sep 15]. Available from: https://pure.mpg.de/rest/items/item_2346268/component/file_2346267/content
9. LeCun Y, Bengio Y, Hinton G. Deep Learning. Nature. 521. 436-44. 10.1038/nature14539. 2015 [cited 2022 Sep 15]. Available from: https://www.researchgate.net/publication/277411157_Deep_Learning
10. Wang M, Deng W. Deep Face Recognition: A Survey. Neurocomputing [Internet]. 2021 Mar [cited 2022 Sep 15]. 429; 215-244. Available from: https://arxiv.org/abs/1804.06655
11. Tesla.com: Artificial Intelligence & Autopilot. c2022 [cited 2022 Sep 15]. Available from: https://www.tesla.com/AI
12. Ramesh A, Pavlov M, Goh G, Gray S, Voss C, Radford A, Chen M, Sutskever I.. Zero-shot text-to-image generation [Internet]. arXiv [cs.CV]. 2021 [cited 2022 Sep 15]. Available from: https://arxiv.org/pdf/2102.12092.pdf
13. Rajpurkar P, Irvin J, Zhu K, Yang B, Mehta H, Duan T, et al. CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning [Internet]. arXiv [cs.CV]. 2017 [cited 2022 Sep 15].. Available from: https://arxiv.org/abs/1711.05225
14. Cohen JP, Hashir M, Brooks R, Bertrand H. On the limits of cross-domain generalization in automated X-ray prediction. Proceedings of Machine Learning Research 1-13, 2020. 2020 [cited 2022 Sep 13]. Available from: https://arxiv.org/pdf/2002.02497.pdf
15. Khan A, Khan SH, Saif M, Batool A, Sohail A, Khan MW. A Survey of Deep Learning Techniques for the Analysis of COVID-19 and their usability for Detecting Omicron. 2022 [cited 2022 Sep 15]. Available from: https://arxiv.org/ftp/arxiv/papers/2202/2202.06372.pdf
16. Fusco R, Grassi R, Granata V, Setola SV, Grassi F, Cozzi D, Pecori B, Izzo F, Petrillo A. Artificial Intelligence and COVID-19 Using Chest CT Scan and Chest X-ray Images: Machine Learning and Deep Learning Approaches for Diagnosis and Treatment. J Pers Med. 2021 Sep 30 [cited 2022 Sep 15];11(10):993. doi: 10.3390/jpm11100993. PMID: 34683133; PMCID: PMC8540782. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8540782/
17. Diaz-Escobar J, Ordóñez-Guillén NE, Villarreal-Reyes S, Galaviz-Mosqueda A, Kober V, Rivera-Rodriguez R, et al. (2021) Deep-learning based detection of COVID-19 using lung ultrasound imagery. PLoS ONE 16(8): e0255886. 2021 Aug 13 [cited 2022 Sep 16]. Available from: https://doi.org/10.1371/journal.pone.0255886
18. Giełczyk A, Marciniak A, Tarczewska M, Lutowski Z. Pre-processing methods in chest X-ray image classification. PLoS One [Internet]. 2022;17(4):e0265949. Available from: http://dx.doi.org/10.1371/journal.pone.0265949
19. Radiologyinfo.org. Pneumonia [Internet]. [cited 2022 Oct 10]. Available from: https://www.radiologyinfo.org/en/info/pneumonia
20. Bell D, Jones J. Pneumonia (summary). In: Radiopaedia.org. Radiopaedia.org; 2015 [cited 2022 Oct 10]. Available from: https://radiopaedia.org/articles/pneumonia-summary?lang=us
21. Koo HJ, Lim S, Choe J, Choi S-H, Sung H, Do K-H. Radiographic and CT features of viral pneumonia. Radiographics [Internet]. 2018;38(3):719–39. [cited 2022 Oct 10]. Available from: http://dx.doi.org/10.1148/rg.2018170048
22. Kafritsas N. Installing PyTorch on apple M1 chip with GPU acceleration [Internet]. Towards Data Science. 2022 [cited 2022 Dec 12]. Available from: https://towardsdatascience.com/installing-pytorch-on-apple-m1-chip-with-gpu-acceleration-3351dc44d67c

Questions?

Thank You!