On the Improvement of COVID-19 Pneumonia Classification on Chest Radiographs through Transfer Learning

University of Connecticut - Summer Research
June - August 2021

By: Grace Liu
# TABLE OF CONTENTS

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>INTRODUCTION</td>
</tr>
<tr>
<td>02</td>
<td>METHODS</td>
</tr>
<tr>
<td>03</td>
<td>RESULTS</td>
</tr>
<tr>
<td>04</td>
<td>CONCLUSION</td>
</tr>
</tbody>
</table>
INTRODUCTION

COVID-19 has caused over 4 million deaths worldwide... how can we diagnose it with machine learning?

Specific Issue: Identifying viral COVID-19 pneumonia on chest radiographs

Kaggle Competition: SIIM-FISABIO-RSNA COVID-19 Detection
DATA OVERVIEW

N (Negative)

1612

T (Typical)

2717

I (Indeterminate)

1003

NT (Atypical)

452
METHODS

- Python, Tensorflow, Keras
- Lopsided data distribution → a model built from scratch is expensive and prone to overfitting
  - Using transfer learning
  - Def: Features learned on one problem are used for another related problem
VGG16 BASIC MODEL

- Using Keras’s pre-trained models and adding classifier layers
- Mismatch between medical file format + model default input sizes
  - 1 channel vs RGB
- Broadcasting
- Very low accuracy ~41% & overfitting
- Simplifying by focusing on binary classification (typical vs atypical)
DATA AUGMENTATION

- **‘equalize_adaphist’**
  Contrast Limited AHE: local contrast enhancement

- **random flip**
  Flipping the image horizontally/vertically

- **random zoom**
  Magnifies the image by a random factor

- **random contrast**
  Adjusts contrast by a random factor
AUGMENTATIONS IN CHANNELS

- Based on previous success with random contrast, we decided to augment the color channels.
- Instead of broadcasting, applying RandomContrast(0.15) to the 2 other channels.
x1 = inputs
x2 = keras.layers.experimental.preprocessing.RandomContrast(0.15)(x1)
encoded_image = keras.layers.experimental.preprocessing.RandomContrast(0.15)(x1)
x = tf.keras.layers.Concatenate(axis=-1)([x1, x2, x3])
TESTING PRE-TRAINED MODELS

- **VGG16**
  - Original experiment

- **Inception_V3**
  - Keras Model

- **ResNet50**
  - Keras Model

- **InceptionResNetV2**
  - Combination of Inception + ResNet

- **EfficientNetB7**
  - B0-B7 Models, 7 is best accuracy
ENSEMBLED MODEL

InceptionResNetV2
Features extracted with this model

EfficientNetB7
Combined with features extracted from this model

RESULT
Significant improvement in accuracy
def make_transfer_inceptionresnet_efficient_model(image_size, num_classes):
    base_model1 = keras.applications.EfficientNetB7(
        weights="imagenet",  # Load weights pre-trained on ImageNet.
        input_shape=(600, 600, 3),
        include_top=False,
    )  # Do not include the ImageNet classifier at the top.
    base_model1 = add_prefix(base_model1,'b1')

    base_model2 = keras.applications.InceptionResNetV2(
        weights="imagenet",  # Load weights pre-trained on ImageNet.
        input_shape=(299, 299, 3),
        include_top=False,
    )  # Do not include the ImageNet classifier at the top.
    base_model2 = add_prefix(base_model2,'b2')
### Model Summary

**Model:** "model_5"

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
<th>Connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_6 (InputLayer)</td>
<td>([None, 600, 600, 1])</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>random_contrast_4 (RandomContrast)</td>
<td>([None, 600, 600, 1])</td>
<td>0</td>
<td>input_6[0][0]</td>
</tr>
<tr>
<td>random_contrast_5 (RandomContrast)</td>
<td>([None, 600, 600, 1])</td>
<td>0</td>
<td>input_6[0][0]</td>
</tr>
<tr>
<td>concatenate_2 (Concatenate)</td>
<td>([None, 600, 600, 3])</td>
<td>0</td>
<td>input_6[0][0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>random_contrast_4[0][0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>random_contrast_5[0][0]</td>
</tr>
<tr>
<td>resizing_1 (Resizing)</td>
<td>([None, 299, 299, 3])</td>
<td>0</td>
<td>concatenate_2[0][0]</td>
</tr>
<tr>
<td>tf.math.truediv_1 (TFopLambda)</td>
<td>([None, 299, 299, 3])</td>
<td>0</td>
<td>resizing_1[0][0]</td>
</tr>
<tr>
<td>tf.math.subtract_1 (TFopLambda)</td>
<td>([None, 299, 299, 3])</td>
<td>0</td>
<td>tf.math.truediv_1[0][0]</td>
</tr>
<tr>
<td>b1efficientnetb7 (Functional)</td>
<td>([None, 19, 19, 2560])</td>
<td>64097687</td>
<td>concatenate_2[0][0]</td>
</tr>
<tr>
<td>b2inception_resnet_v2 (Function)</td>
<td>([None, 8, 8, 1536])</td>
<td>54336736</td>
<td>tf.math.subtract_1[0][0]</td>
</tr>
<tr>
<td>global_average_pooling2d_2 (Glo)</td>
<td>([None, 2560])</td>
<td>0</td>
<td>b1efficientnetb7[0][0]</td>
</tr>
<tr>
<td>global_average_pooling2d_3 (Glo)</td>
<td>([None, 1536])</td>
<td>0</td>
<td>b2inception_resnet_v2[0][0]</td>
</tr>
<tr>
<td>concatenate_3 (Concatenate)</td>
<td>([None, 4096])</td>
<td>0</td>
<td>global_average_pooling2d_2[0][0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>global_average_pooling2d_3[0][0]</td>
</tr>
<tr>
<td>flatten_1 (Flatten)</td>
<td>([None, 4096])</td>
<td>0</td>
<td>concatenate_3[0][0]</td>
</tr>
<tr>
<td>dense_1 (Dense)</td>
<td>([None, 2])</td>
<td>8194</td>
<td>flatten_1[0][0]</td>
</tr>
</tbody>
</table>

**Total params:** 118,442,617  
**Trainable params:** 8,194  
**Non-trainable params:** 118,434,423
RESULTS
Discussing final accuracy obtained and conclusions
### Table 1: A comparison of results from all different models

<table>
<thead>
<tr>
<th>Model Name and Description</th>
<th>Classes Compared</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16 model</td>
<td>N, T, I, NT</td>
<td>41%</td>
</tr>
<tr>
<td>VGG16 model with Contrast Limited AHE (CLAHE)</td>
<td>N, T, I, NT</td>
<td>39%</td>
</tr>
<tr>
<td>VGG16 model</td>
<td>T, NT</td>
<td>69%</td>
</tr>
<tr>
<td>VGG16 model with variety of data augmentation functions</td>
<td>T, NT</td>
<td>66%</td>
</tr>
<tr>
<td>VGG16 model with channel augmentation</td>
<td>T, NT</td>
<td>73%</td>
</tr>
<tr>
<td>Inception_V3 model with channel augmentation</td>
<td>T, NT</td>
<td>68%</td>
</tr>
<tr>
<td>ResNet50 model with channel augmentation</td>
<td>T, NT</td>
<td>67%</td>
</tr>
<tr>
<td>InceptionResNetV2 with channel augmentation</td>
<td>T, NT</td>
<td>76%</td>
</tr>
<tr>
<td>EfficientNetB7 model with channel augmentation</td>
<td>T, NT</td>
<td>76%</td>
</tr>
<tr>
<td>InceptionResNetV2 + EfficientNetB7 model with channel augmentation</td>
<td>T, NT</td>
<td>81%</td>
</tr>
</tbody>
</table>
ACCURACY CURVE COMPARISON

Earliest Attempt (VGG16)

Final Attempt (Ensembled Model)
METHOD SUCCESSES AND FAILURES

1. Augmenting the color channels ~7% increase in accuracy from basic data augmentation
2. Combining features extracted from 2 models ~5% increase in accuracy

1. Basic data augmentation methods - Filters, basic manipulation, etc.
2. Broadcasting
3. Using single model
FUTURE IMPROVEMENTS

● Experimenting further with the ensemble model -- would combining features from more than 2 models be more successful?
● I used equal sample sizes from each class at the beginning
  ○ Making full use of all the images and adding more weights
● Finally, testing out additional augmentation strategies
REFERENCES


Kaggle Competition:
ACKNOWLEDGEMENTS

● Professor Jinbo Bi for advising this project
● Professor Shengli Zhou for organizing this summer research opportunity
● All the other professors and graduate students who presented and provided suggestions
● Jian Liu for also advising this project
THANKS!

Any questions?
SEO & Marketing Icons