Using Deep Learning to Detect COVID-19 From Lung CT Scans

Eddy Mina*, Isaac Dekine*, Qiang Hu*, Ziwei Wu*

Abstract

COVID-19, a contagious respiratory illness which started in 2019, quickly caused a wide scale pandemic impacting billions of people and strained healthcare systems globally. Deep learning models have demonstrated a robust ability to accurately perform well on image classification tasks which can play a vital role in accelerating diagnosis. We implemented several existing state-of-the-art Deep learning models in order to compare their inference time and ability to learn on a small dataset of pulmonary CT scans. In this report, we assess the impact of different training approaches and hyperparameters on the efficacy of those models, and interpreted our findings. EfficientNet was found to be the best performing model, with an overall accuracy of 97.3% and transformer based models such as LeViT also achieving over 97% accuracy. Our strong results obtained using only a few thousand images shows promise in training similar models with less computing resources.

1. Introduction

1.1. Background

The SARS-CoV-2 virus or COVID-19 is a contagious respiratory illness causing the ongoing pandemic with first reports beginning as early as November 2019 [1]. Currently, diagnosis of the virus is primarily done using a PCR virology nasal swab assay [2]. The PCR swab has a broad false negative rate ranging from 1-30% caused by, but not limited to, poor specimen collection, testing too early, low analytic sensitivity, and low viral load [3]. Furthermore, results are only known after 2-5 days from testing. As a result, guidance from the World Health Organization has strongly pushed for repeated testing including using CT scans as complementary testing, especially for high risk individuals. Although CT scans do not diagnose the illness directly, they can quickly find respiratory abnormalities caused by illness such as the flu, pneumonia, and COVID-19 [4].

*Names ordered alphabetically, all authors made significant contributions

1.2. Motivations

In the past decade, Deep learning has been shown to achieve human like performance on tasks such as image recognition [5]. We are interested in leveraging this powerful development to help diagnose COVID-19 using CT scans. Our primary motivation is to use and compare state of the art deep image classifiers to potentially automate COVID-19 diagnoses on a larger scale. Since neural networks can generalize, we also hope to identify possible future variants which virology tests may fail to.

With billions across the world in constant need for testing, scaling and quickening testing has the potential to keep patients and doctors well informed on disease progression, reducing risks for complications including mortality. Likewise, Deep learning models can run on batch anywhere in the world, including regions without many skilled radiologists or technicians to interpret the scans. These models can also offer second opinions to overworked healthcare staff to reconsider any overlooked details or patient information.

1.3. Data

The dataset used for our large scale classification task is from COVID-Net Open Source Initiative, a global initiative dedicated to fighting the continuous COVID-19 pandemic with open source, open access, curated benchmark medial images and tailored deep neural network models [6]. Specifically the data used contains 201,103 CT images of three different patient conditions: novel coronavirus pneumonia, common pneumonia, and normal controls (Table 5). The distribution of the dataset is skewed towards COVID-19 patients, with 57.23% having COVID-19, 17.73% with Pneumonia, and the remaining 25.04% are normal. The remaining metadata showed a majority of the data originates from China (over 68%), an equal male-female distribution, and a normal distribution of age averaging 51.01 ± 16.7 years. For our implementation we directly utilized the raw CT scans. Each CT scan is a single axial grayscaled cross sectional view of both lungs taken either automatically or by technician for each patient stored as a png file.

2. Approach

Using publicly available data, we performed a range of experiments to determine the best Deep learning approach for classifying CT scans to quickly assess whether the patient has pneumonia, COVID-19, or neither. As previous work has shown [7] [8] [9], various models can achieve fairly good performance, however the results were focused on result proof-of-concept. In our project, we strived to take a more rigorous exploration and build a better understanding of model performance in two ways: quantitatively comparing measured performance across state-of-the-art models and the approaches to training them, and qualitatively to view the model feature-space and be confident that they are learning meaningful representations.

To do this, we first started with a large dataset of CT scan images and created a unified framework to pre-process and ingest the data. Second, we implemented an approach to quickly perform hyperparameter tuning in order to train and test a variety of Deep learning models and evaluate their performance. Finally, we developed analysis and plotting software to assess the overall model results and to visualize the feature space.

2.1. Implementation

Because the dataset was very large for consumer-grade processing, and our computing resources were limited, our first step was to sub-sample the data to a manageable level. The original dataset consisted of 30 GB of CT scan image data, comprising approximately 195,000 monochrome images of varying dimensions (mostly 512x512 or 630x630), which were also preprocessed with augmentation and normalization. From this, we downsampled to create two smaller sets: a 10% subset and a 1% subset (3 GB and 300 MB, respectively), taking uniformly random samplings of the originally provided training, testing, and validation data portions. We did not perform any data-balancing steps at this stage, opting instead to utilize focal loss to compensate for the imbalance, and to evaluate the effect of the imbalance on validation performance. Next, we used transforms to augment the training data on-the-fly, by performing random horizontal and vertical flip (50% probability), and random rotation by up to 30 degrees. The original dataset provided bounding box annotations for the body regions in the CT scans, however for better generalization of the models, we opted to not use this information, instead we applied center crops with different sizes, and then resized to a fixed size of 244x244. Finally, we applied data normalization, to align with the mean and standard deviation of ImageNet (when using pre-trained models), or to align with the overall dataset (when training from scratch).

We then implemented a variety of state-of-the-art Deep learning models, categorized as convolutional-based models (CNNs) and transformer-based models, known as vision transformers (ViTs). To assess the impact of transfer learning, we utilized pre-trained versions of these models (pre-trained on ImageNet), as well as non-pretrained models. We explored two flavors of pre-training: first, in which we froze the entire network except for the final layer (typically a fully-connected output layer), which we replaced and trained from scratch during training in order to learn the mapping to output classes; or second, in which we replaced that final layer, but did *not* freeze the rest of the network, such that training would also perform gradient descent updates to the entire network.

Next, we utilized a Python framework (Raytune) [10] to assist with hyperparameter tuning of the models. We performed independent tuning over each of the models, to determine the best hyperparameter set for each model. We also ran this tuning over other experiments. The experiments we performed specifically were:

- Center-cropping size: 340x380, or 480x480
- Loss function: Focal Loss vs Cross-Entropy
- Input channel approach, discussed later in section 2.3
- Model pre-training: no pre-training (train from scratch), pre-train but only update final layer, pre-train but allow updates for entire network.

Finally, we created software to take results across all of the experiments performed, and evaluate their performance. In addition, we implemented GradCAM to evaluate how the models learned to perform the classification.

2.2. Models

We focused on state-of-the-art convolutional model architectures, in particular EfficientNet [11], InceptionNet [12], MobileNet [13], and ResNet [14]. EfficientNet is an architecture that allows for principled scaling of the network (in image dimension, as well as in network width and depth) in an efficient, unified manner. InceptionNet is a CNN architecture designed to account for input scale variability by containing multiple convolutional filter sizes at each layer of the network. This is particularly relevant in our experiments, because the size of lungs, as well as of the lesions indicative of pneumonia or COVID, may vary significantly from patient to patient. MobileNet is a CNN architecture designed for computational efficiency in mobile devices using a number of optimizations including separable convolutions. Since we envision that rapid diagnosis of images may occur on some sort of embedded or mobile device by medical professionals, understanding how detection performs on this model is important. Finally, ResNet is an architecture that utilizes residual network connections to deal with the vanishing gradient problem present in deep CNNs.

Vision Transformers (ViTs) are a relatively new development [15], and we tested two example ViT architectures: Data-efficient Image Transformers (DeiT) [16], and LeViT [17]. Data-efficient Image Transform (DeiT) features a student-teacher setup and utilize distillation tokens to allow the model to learn from the teacher output while remaining complementary to the actual class embedding. LeViT is an architecture that combines a convolutional architecture and a transformer architecture, first pre-processing the input with ResNet-50, and later including shrinking-attention blocks in the transformer section which utilize subsampling and convolutions to reduce the dimensionality further.

2.3. Challenges

Two main challenges arose during the course of this research: utilizing models designed for RGB images with grayscale data, and computational strain given the size of the dataset and the models.

First, we applied a number of existing deep image learning algorithms, including pre-trained models in order to examine the effect of transfer learning. These models were designed for three-channel (RGB) image data input, however CT scans are single-channel data, as they measure wave attenuation at a single wavelength. As a result, we had to devise an approach for utilizing these pre-trained models with single-channel input data. We derived three approaches to this problem:

- Use the weights for one of the 3 possible input channels, and modify the input layer of the model to only use this set of weights (convert from 3xN to 1xN)
- Average the weights across the 3 input channels, and modify the input layer of the model to use this averaged set of weights (again, converting a 3xN layer to 1xN)
- Replicate the input data into 3 identical channels (converting from greyscale to RGB), which does not require modifying the input layer of the model

We performed experiments across these possible approaches to determine which one was the most effective, and treated this as another "hyperparameter" to tune when identifying the best model parameters.

Second, the size of the dataset yielded a significant challenge in performing experiments. One major problem that we encountered was computer resource limitations when training, even with 10% of the dataset. Three out of four team-members ran into errors with local machine or Google GVM instance likely due to GPU memory limitation. In one instance, a team member had to reduce the batch size significantly (16 samples) to successfully run the models, and even despite that, the resulting training epoch time was measured in hours. To mitigate this problem, we also downsampled the data further and created a 1% dataset version for testing purposes. The final models were trained and evaluated with 10% dataset using a two NVIDIA Tesla A100 GPU server. Ultimately, this allowed us to explore performance using a smaller dataset, which was not performed in prior work.

3. Experiments and Results

Four CNN models and two vision transformer models were implemented using the pytorch v1.10 [18] and torchvision v0.11 packages. For each model, four groups of hyperparameters were explored and tuned to develop the best models for detecting COVID-19 from CT scan images. Therefore, we experimented on 6 models, performing a total of 60 hyperparameter combination trials per model, summing to 360 trials in total. The best hyperparameter set for each model was determined by the highest accuracy on a fixed validation dataset.

3.1. Comparison performance of best models

To benchmark the performance of our 6 models quantitatively, classification metrics, including Precision, Recall, F1 score and Accuracy were computed for each class on the test dataset. The six best models from hyperparameter tuning are selected based on the highest accuracy on the validation dataset. Table 1 presents the performance metrics of the six models on the test dataset. All six models classified the three classes of lung disease successfully and achieved very high overall accuracy (> 0.9). The EfficientNet achieved the highest overall accuracy (0.973) on the test dataset.

The six models performed differently on the three target groups. At the task of classifying COVID-19, EfficientNet achieved the highest F1-score (0.954), implying that it is the most balanced model in terms of both precision and recall. However, the highest precision and recall were LeViT (0.946) and ResNet (0.960) respectively, thus ResNet had the highest sensitivity for COVID-19, which is important if we want to maximize its detection. For identifying normal lungs, EfficientNet obtained the highest precision (0.992), while the LeViT model achieved the highest recall (0.995) and MobileNet reached the highest F1-score (0.988). For the pneumonia group, ResNet obtained the highest precision (0.989) and F1-score (0.970), but the highest recall was EfficientNet (0.991).

3.2. Comparison of Deep learning process

Training processes were explored for better understanding and optimizing these models. Learning curves were plotted for two high performing models, EfficientNet and LeViT, a CNN and vision transformer model respectively, in figure 1. The figure presents the loss values and accuracies for both training and validation datasets over the training epochs. The accuracy and loss curves for EfficientNet show the model fluctuates before 5 epochs and converges in the end. The LeViT model shows more stable accuracy increasing in the training process and its loss curves suggest it requires more epochs and smaller learning rates to reach better performance.

We compared the best model accuracy values for the six models on the train, validation and test dataset (Fig-

Table 1. Performance comparison of best models on CT scan test dataset

Model	Label	Precision	Recall	F1-score	Overall Accuracy
ResNet 152	Normal	0.973	0.984	0.979	0.969
	Pneumonia	**0.989**	0.951	**0.970**	
	COVID-19	0.936	**0.960**	0.948	
Inception V3	Normal	0.991	0.983	0.987	0.969
	Pneumonia	0.967	0.959	0.963	
	COVID-19	0.926	0.952	0.939	
MobileNet V3	Normal	0.987	0.988	**0.988**	0.964
	Pneumonia	0.943	0.959	0.951	
	COVID-19	0.944	0.919	0.931	
EfficientNet B7	Normal	**0.992**	0.981	0.986	**0.973**
	Pneumonia	0.943	**0.991**	0.966	
	COVID-19	0.943	0.933	**0.954**	
DeiT p16	Normal	0.931	0.969	0.950	0.911
	Pneumonia	0.876	0.907	0.891	
	COVID-19	0.913	0.794	0.849	
LeViT 384	Normal	0.979	**0.995**	0.987	0.970
	Pneumonia	0.973	0.953	0.963	
	COVID-19	**0.946**	0.940	0.943	

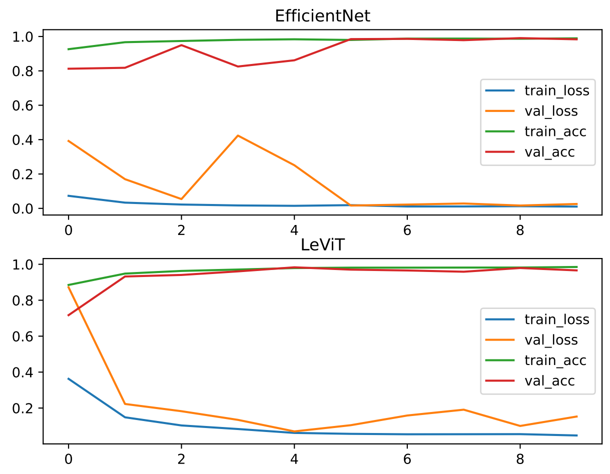

Figure 1. Learning curves of EfficientNet and LeViT

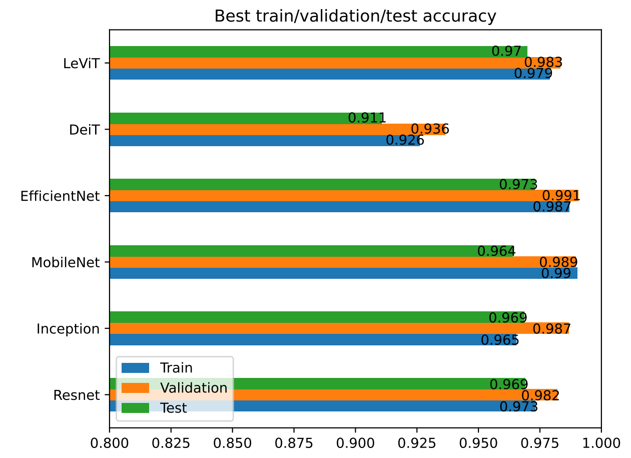

Figure 2. Training, validation and test accuracy from best models

ure 2). The barplots show the models' performance differences across the datasets. The MobileNet obtains the highest training accuracy and the EfficientNet achieves the highest validation and test accuracy. Comparing the results across the three datasets, most of the models show small accuracy differences (about 0.1%), except MobileNet drops about 0.3% in the test dataset. The results suggest most of the models have good generalization performance.

3.3. CNN vs. Vision Transformer

Two groups of Deep learning architectures were applied to CT scan datasets, CNN and Vision Transformer. Vision transformers are now competitive alternatives to CNN models that outperform the current CNNs in terms of computational efficiency and accuracy. Table 1 shows the best 2 models, which were EfficientNet (CNN) and LeViT (transformer). Table 2 lists the best accuracy of models pre-trained on ImageNet, number of parameters and average training time per epoch on the COVID-19 dataset. LeViT achieved similar accuracy to EfficientNet, with roughly half as many parameters and much faster runtime.

3.4. Hyperparameter sensitivity

We explored the effect of various hyperparameters on performance of the six models to obtain the best model, specifically, the image channel to use, the image size to crop, the loss function and the pre-trained initial weights with or without freezing feature extracting layers, as listed

Table 2. CNN vs. Vision Transformer

Model	Acc@5	parameters	time (s)
ResNet 152	94.05	60192808	270.10
Inception V3	93.45	27161264	238.70
MobileNet V3	91.34	5483032	243.08
EfficientNet B7	96.91	66347960	336.90
DeiT p16	95.60	86567656	305.70
LeViT 384	96.00	39128836	196.30

in the table 3. The grid search algorithm implemented by the Ray Tune package was used to compare the performance among a total of 60 hyperparameter configurations for each model. Other parameters are chosen and fixed by experience, such as learning rate (0.001), batch size (64), image input size (224 or 299), number of training epochs (10), Adam optimizer and its parameters and Focal Loss parameters. The best configurations from the highest validation accuracy for each model are listed in table 3.

3.4.1 Pre-trained initialization weights vs. Transfer Learning vs. Learning from scratch

We tested the impact of three methods of model training on task performance. First, the configuration with pre-trained "Yes" and feature extracting "No" means to initialize the model parameters with the corresponding pre-trained model from the ImageNet dataset on 1000 categories. Transfer learning also uses pre-trained initialization weights and freezes the convolutional feature-extracting layers (pre-trained "Yes" and feature-extracting "Yes"), then only trains the last fully connected layers for classification. Last, we also tried to learn from scratch with random initialization parameters (pre-trained "No" and feature-extracting "No"). Table 3 shows that all the best models are trained by pre-trained initialization weights under current setup, which indicates that features extracted from totally different RGB image sets still work well on the grayscale medical images. However, learning from scratch actually could achieve better performance with fine tuning and enough training time because the pre-trained initialization could lead to local optima [19].

3.4.2 Medical grayscale image vs. RGB image

It is still an open question of how to apply current 3 channel CNN architectures to grayscale medical images. Specifically, there are RGB channels of parameters from pre-trained models but only one channel in medical image. To address this challenge, we investigated different methods to utilize the 3 channel data, including just one of the RGB channels, aggregating weights by average, and repeating grayscale channel to 3 channel image. Table 3 shows that different architectures work better with different channels of pre-trained parameters, but most of the models work well with just one channel of input except DeiT. Therefore, the results suggest we should keep one channel grayscale image as input to train the models.

3.4.3 Crop size

Two different scales of center crop sizes were tested because only the center lung areas provide meaningful information for the target groups. Most of the models take larger crop sizes because of more information as inputs. Ideally, if we can segment the lung region for training, the model can do much better by focusing on the lesion regions. We have tried intensity-based fast segmentation to extract lung areas. However, this method failed to segment the CT scans with unclear lung boundaries.

3.4.4 CE vs. FL

We investigated two loss functions, cross entropy (CE) and focal loss (FL) [20]. The focal loss could help to assign more weights to the imbalanced target groups as in our training dataset. We have a large enough number of images for each group in the training dataset despite having imbalanced classes, which might explain why both CE and FL work well in different models (Table 3).

3.5. Qualitative analysis

To qualitatively analyze how models learned to classify target groups from the CT scans, we use Grad-Cam [21] to generate class-special heatmaps on original images. The Grad-Cam can produce visual explanations for decisions from deep networks by highlighting localization maps based on gradient of weights. We present Grad-Cam heatmaps from EfficientNet and DeiT for two cases of CT scans of Pneumonia and COVID-19 respectively in figure 3. The top two figures are from EfficientNet and the bottoms are from DeiT. The left two figures are from the same COVID-19 case and the right twos are from the same Pneumonia case. The two cases both show typical pneumonia-like lesions: ground glass opacities and consolidation. The COVID-19 case shows reticulations in bilateral posterior basal lung, but the Pneumonia show more extensive lesions. The Grad-Cams highlight most of the lesion locations correctly. The two models show different patterns in the heatmaps. The EfficientNet highlights the lesion area more intensively, however the DeiT marks multiple regions to make decisions.

4. Discussion

4.1. Conclusion and Broader Impacts

We examined the performance difference between CNN based and transformer based models in COVID-19 clas-

Table 3. Best hyperparameter configurations

Model	Pre-trained	Feature extraction	Image channel	Crop size	Loss function
ResNet 152	Yes	No	R	(480, 480)	CE
Inception V3	Yes	No	average	(480, 480)	FL
MobileNet V3	Yes	No	R	(340, 380)	CE
EfficientNet B7	Yes	No	G	(480, 480)	FL
DeiT p16	Yes	No	RGB	(340, 380)	FL
LeViT 384	Yes	No	R	(480, 480)	CE

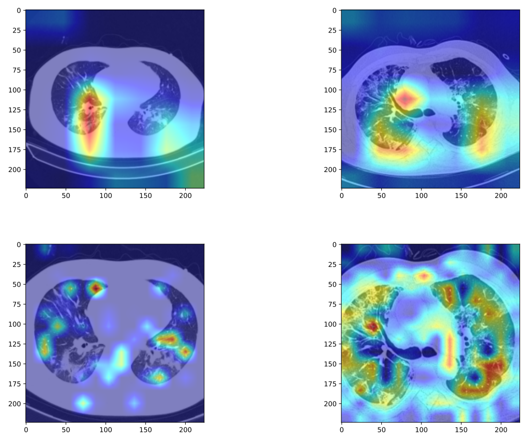
Figure 3. Visualization of COVID-19 (left) and Pneumonia (right) cases using Grad-Cam heatmaps from EfficientNet (top) and DeiT (bottom).

sification. Although EfficientNet is the best overall performing model in classifying COVID 19 using CT images based, transformer based models showed great performance as well. LeViT only trailed behind EfficientNet in terms of overall accuracy. There is great potential for application of transformer models in vision tasks.

Medical grayscale images, especially radiological images from X-ray, CT and MRI etc, play a crucial role in assisting clinical diagnosis and therapy. However, most of the current deep networks are developed and pre-trained on RGB images. Many studies simply repeat single channels to create a 3 channel equivalent to adopt existing neural network architectures, such as the COVIDNet-CT [7]. Our study explored and accessed different methods to apply 3 channel network models to single channel medical images. Our findings suggest single channel images are less computationally expensive and sufficient to achieve SOTA accuracy.

We have shown that we are able to detect COVID 19 with high accuracy using a 10% dataset compared to prior work. This is important because in healthcare, large datasets of medical images are hard to access due to reasons such privacy, compliance etc. This allows research to generate models for classifying COVID 19 when there are limited number of data. Another benefit of achieving equivalent performance using only 10% dataset is faster training speed. This opens up the potential to train these models using less powerful devices e.g medical frontline in remote regions.

4.2. Future Improvements

In our comparison between pretrained model vs non-pretrained models. We found that pretrained models consistently outperformed non-pretrained models. We suspect the reason for this is that non-pretrained models might require more epochs to optimize. Limiting the training to 10 epochs might have caused non-pretrained models to underfit. However, we believe that by using more epochs, non-pretrained models can achieve comparable performance and it's something we would like to explore in future.

We didn't perform grid search all the possible hyper parameters e.g learning rate, batch size since adding more hyperparameters in grid search is exponential in computational complexity. This would prevent us from obtaining results within project timeline. We would like to explore other searching algorithms in the future such Bayesian Optimization [22] or Population Based Training [23] that don't perform exhaustive search on hyper parameter space and can shorten tuning time.

We did experiments with various data augmentation techniques including center cropping and random horizontal / vertical flip. Random crop was a technique we didn't think of while conducting the experiments. Laskin et al [24] demonstrated that random cropping is the single most effective technique in their study on improving performance for pixel based reinforcement learning problems. Random cropping could be something we can try in future to see if we can improve performance further.

Lastly, radiological images, such as CT and MRI scans, are actual volumetric one channel and 3D data with extra depth dimension for space information. It is still very challenging to apply current neural network architectures to 3D medical data. However, exploring ways to fully take advantage of 3D medical data in Deep learning would be an interesting future research direction to pursue.

Student Name	Contributed Aspects
Qiang	Data preparing, Model implementation, Model tuning, training and evaluation, Generating tables and figures for report, Grad-Cam, Report writing
Isaac	Model implementation, plotting and comparison function implementation, automated data sharing (downloading) of large data subsets, report writing and proofreading
Eddy	Model implementation, Training Code Implementation, Code review, Managing Project Timeline and maintaining One Drive folder, Report Writing and proof reading
Ziwei	Hyperparameter tuning implementation, tuning visualization and data reporting implementation, plot function implementation, lead weekly group meeting and manage project timeline, report writing

Table 4. Contributions of team members. For more details on contribution please see Work Division, section 5

5. Work Division

Summary of contributions provided by each team member can be found in table 4, and in more detail in the bullets below.

5.1. Qiang Hu

- Data preparation: Downloaded the COVIDxCT 2A full datasets and down-sampled to 1% and 10% subsets (images, meta tables and down_sampling.py in the down-sampled dataset A2.1 and A2.2).
- Model implementation: Implemented data augmentation and main training scripts (dataset.py, load_data.py, train.py, main.py, loss.py). Implemented ResNet, DistillViT, DeiT, LeViT (Models/resnet.py, Models/DistillViT.py, Models/deit.py, Models/LeViT.py).
- Model tuning, training and evaluation: Performed all the tuning and training experiments on local cluster server. Design and set up all the hyperparameter configures (config/*.py). Summarized the best models based on performance on the validation dataset (evaluation.py, best_model.py). Performed evaluation on test dataset (evaluation.py).
- Generating tables and figures for report: Generated table 1, 2 and 3 to summarize our results (evaluation.py, stats.py). Generated figures 1, 2 and 3 (1, 2 from stats.py, 3 from grad_cam.py).
- Implemented Grad-Cam. Implemented Grad-Cam and generated examples for EfficientNet and DeiT (figure3 and grad_cam.py).
- Report writing: Wrote section 3 Experiments and Results, section 4 about medical images (4.1p2, 4.2p4).
- Code contribution: 30 commits 1,922 ++ 455 –

5.2. Isaac Dekine

- Model implementation: implemented the MobileNet CNN model (models/mobilenet.py). Resolved issues with implementation of ResNet and InceptionNet (models/inception.py and models/resnet.py) regarding application of pre-trained weights, and revising output layer for our 3-class problem, instead of ImageNet output classes. Implemented experiment to allow selection of input channel weights in pre-trained model, in each of the CNN models (which single channel weights to use, or average of three).
- Data downloading / decompression: implemented file fetch utility to download and automatically decompress downsampled datasets that we stored in OneDrive (get_datafile.py)
- Implemented result loading / plotting function (cross_compare.py) to load results of all experiments across all models / hyperparameters, and perform customizable filtering to show results of various experiments (e.g. to compare focal loss vs CE loss). Assessed model efficacy and generated figures.
- Wrote and helped to revise significant sections of the report.

5.3. Eddy Mina

- Model Implementation: implemented the following file structures under the ”Models” directory including inception.py, efficienet.py, and mobilenet.py. Each python file has either a custom implemented parameters such as loss function (inception with auxiliary logits) or uses a shared function such as loss function written in the utils.py.
- Implemented generic loss function based on provided criterion and set_parameter_requires_grad in utils.py as well as the general model constants in the constants.py file. Implemented train loop portion of train.py including gradient propagation and storing model metrics as a DataFrame. Likewise implemented if/else code for the model selection in the main.py and evaluation.py as part of model for importing respective functions and constants during training and evaluation.
- Trained mobilenet.py to obtain premliminary results to validate model efficacy on 1% on data. Likewise validated grad cam results in grad_cam.py of the Efficientnet and ViT models
- Tracked and logged project progress on github project

page creating tasks, writing meaning minutes, general ideas and proposals and storing information on OneDrive.

- Paper Report implementation. Did background research on illness, current state of the art research and methods, and background information on dataset initiative and described this information in the introduction in section 1. Likewise contributed to writing the final abstract in the beginning of the paper.

5.4. Ziwei Wu

- Implemented Hyperparameter tuning: investigated and implemented the code to set up hyperparmaeter tuning using Raytune (see implementations in main.py file and files under config folder and enums folder). Refactored data loading function to work with Raytune (see load_data.py)
- Implemented tuning visualization and data reporting: implemented the code to set up tuning visualization and data reporting using tensorboard (see main.py)
- Implemented plot function to generate learning curve for a single model (see plot.py)
- Lead weekly group meeting and manage project timeline: Set up weekly group meetings, leading the meeting discussions, set timeline for different stages of the project to ensure we finish the project on time
- Report writing: Writing and owning section 4 Discussion. Also contributed to other sections including abstract, and section 2.3 on challenges on limited computing.
- Code contribution: 30 commits 993 ++ 629 –

References

[1] J. Li, S. Lai, G. F. Gao, and W. Shi, "The emergence, genomic diversity and global spread of SARS-CoV-2," *Nature*, pp. 1–11, Dec. 2021. 1

[2] CDC, "Cdc diagnostic tests for covid-19." `https://www.cdc.gov/coronavirus/2019-ncov/lab/testing.html`, Feb. 2020. 1

[3] J. N. Kanji, N. Zelyas, C. MacDonald, K. Pabbaraju, M. N. Khan, A. Prasad, J. Hu, M. Diggle, B. M. Berenger, G. Tipples, and et al., "False negative rate of covid-19 pcr testing: a discordant testing analysis - virology journal," Jan 2021. 1

[4] S. Machnicki, D. Patel, A. Singh, A. Talwar, B. Mina, M. Oks, P. Makkar, D. Naidich, A. Mehta, N. S. Hill, K. K. Brown, and S. Raoof, "The Usefulness of Chest CT Imaging in Patients With Suspected or Diagnosed COVID-19: A Review of Literature," *CHEST*, vol. 160, pp. 652–670, Aug. 2021. Publisher: Elsevier. 1

[5] A. K. G. Inc, A. Krizhevsky, G. Inc, G. I. Profile, I. S. G. Inc, I. Sutskever, G. E. H. OpenAI, G. E. Hinton, OpenAI, O. Profile, and et al., "Imagenet classification with deep convolutional neural networks," Jun 2017. 1

[6] H. Gunraj, A. Sabri, D. Koff, and A. Wong, "Covid-net ct-2: Enhanced deep neural networks for detection of covid-19 from chest ct images through bigger, more diverse learning," 2021. 1

[7] L. Wang and A. Wong, "Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images," 2020. 2, 6

[8] H. Gunraj, L. Wang, and A. Wong, "Covidnet-ct: A tailored deep convolutional neural network design for detection of covid-19 cases from chest ct images," 2020. 2

[9] W. Zhao, W. Jiang, and X. Qiu, "Deep learning for covid-19 detection based on ct images," *Scientific Reports*, no. 11, 2021. 2

[10] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018. 2

[11] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020. 2

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015. 2

[13] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019. 2

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. 2

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. 2

[16] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," 2021. 2

[17] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze, "Levit: a vision transformer in convnet's clothing for faster inference," 2021. 2

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019. 3

[19] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. V. Le, "Rethinking Pre-training and Self-training," *arXiv:2006.06882 [cs, stat]*, Nov. 2020. arXiv: 2006.06882. 5

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *arXiv:1708.02002 [cs]*, Feb. 2018. arXiv: 1708.02002. 5

[21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," *International Journal of Computer Vision*, vol. 128, pp. 336–359, Feb. 2020. arXiv: 1610.02391. 5

[22] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," in *International Conference on Machine Learning*, pp. 1437–1446, PMLR, 2018. 6

[23] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, *et al.*, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017. 6

[24] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *arXiv preprint arXiv:2004.14990*, 2020. 6

A. Appendix

A.1. Link to original dataset page:

https://www.kaggle.com/hgunraj/covidxct

A.2. Link(s) to downsampled datasets

A.2.1 1% dataset

https://gtvault-my.sharepoint.com/:u:/g/personal/idekine3_gatech_edu/Ea_QEMSdsZtLhu_0S6jFE_4Bu-q8_vEZtD2-wpGCl65Irg

A.2.2 10% dataset

https://gtvault-my.sharepoint.com/:u:/g/personal/idekine3_gatech_edu/ESvEK8q0fvVIptjb_PcJzEUBELqJ0ui55B9vF6KfCfRjhw?e=i4kRy8

A.2.3 Dataset Distribution

Table 5. Dataset Training, Validation, and Testing Distribution.

Split	Normal	Pneumonia	COVID-19	Total
Train	35996	25496	82286	143778
Validation	11842	7400	6244	25486
Test	12245	7395	6018	25658

A.3. Code Repo link

The code repo: https://github.com/ziweiwu/dl-health-final-project/

We have attached the codebase as zip file as final project supplement on gradescope. However if you like to access the repo on github, please send an email to zwu373@gatech.edu. We will send you an access invitation link.