MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

A STUDY EXPERIMENTING WITH CNN ENSEMBLE, DEEPMIND PERCEIVER AND GENETIC ALGORITHMS ON COVID DETECTION USING CHEST X-RAYS

By Srinivasaraghavan Seshadhri
R00195470
MSc Artificial Intelligence
2020-2021

Declaration of Authorship

I, Srinivasaraghavan Seshadhri, declare that this thesis titled, 'A STUDY EXPERIMENTING WITH CNN ENSEMBLE, DEEPMIND PERCEIVER AND GENETIC ALGORITHMS ON COVID DETECTION USING CHEST X-RAYS' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Master's Degree at Munster Technological University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- I understand that my project documentation may be stored in the library at CIT, and may be referenced by others in the future.

Signed: Srinivasaraghavan Seshadhri

Date: 01 September 2021

Abstract

The current COVID-19 pandemic has impacted globally with over 183 million cases and 4 million deaths (as of 6 July 2021) [1]. COVID-19 can be misdiagnosed as pneumonia or lung cancer, which with the fast spread in the chest cells, can lead to patient death [2]. Early identification of the disease could help save more patients during this pandemic [3]. Chest x-rays can be used to diagnose and detect Covid-19 [4]. With the huge volume of cases, an assistive diagnostic system could be helpful for radiologists. Extensive research has been conducted by several people using different datasets, architectures, optimization, image processing and feature extraction techniques. Several of these techniques involve CNNs [5-20]. However, use of genetic algorithms, certain ensemble techniques and Google's DeepMind perceiver has not been implemented for this application. In this research, we experiment with said algorithms and evaluate.

Acknowledgement

I would like to pay my sincere gratitude to Dr Ted Scully for his valuable inputs and continuous support throughout the year.

I also thank Munster Technological University for giving me the opportunity to undertake higher studies in my field of choice and perform this research.

I express my very profound gratitude to my parents for their continuous support and encouragement throughout.

Contents

Declaration of Authorship 2
Abstract 3
Acknowledgement 4
Introduction 8
Literature Survey 8
Convolutional Neural Networks 8
COVID-X NET 8
CoroNet 9
ELM Chimp 9
Manta Ray Foraging Optimization 10
Popular CNN Architectures 10
Xception 12
VGG 12
ResNet 12
DenseNet 14
NASNet 14
MobileNet 15
InceptionResNet 16
Genetic Algorithms 16
Neuro-Evolution Augmenting Topologies 16
Transformer Networks 16
Vision Transformer 18
Perceiver 19
Methodology 20
Problem Definition 20
Proposed Approach 20
Dataset 20
Implementation 21
Image pre-processing 21
Baseline CNN 21
Ensemble 22

Genetic algorithm for optimization 22
NEAT algorithm 22
Perceiver 23
Results 23
Convolutional Neural Networks 23
Xception 23
VGG16 24
VGG19 24
ResNet50 25
ResNet101 25
ResNet152 26
ResNet50V2 26
ResNet101V2 27
InceptionV3 27
InceptionResNetV2 28
MobileNet 28
MobileNetV2 29
DenseNet121 29
DenseNet169 30
DenseNet201 30
NASNetMobile 31
NASNetLarge 31
Ensemble 32
Mean Ensemble 32
ELM Ensemble 33
Deeper ANN Ensemble 33
Genetic Algorithm for Optimization 35
Neuro-Evolution for Augmenting Topologies 35
Perceiver 36
Conclusion 37
Convolutional Neural Networks 37
Ensemble 37
Genetic Algorithm 37

NEAT algorithm 38
Perceiver 38
Bibliography 38

Introduction

This research focuses on diagnosing and detecting Covid-19 from chest X-rays using deep learning. Several techniques using convolutional neural networks have been attempted [5-20] such as VGG16, VGG19, ResNet50, ResNet101, ResNet152, Xception, Inception etc. These networks have been trained on different datasets using different optimizers, hyperparameters and techniques. This makes it difficult for us to compare the performance of these algorithms.

In this paper we train some of these networks from scratch on a single dataset with constant hyperparameters and techniques. We also explore other unexplored techniques such as certain ensemble techniques, genetic algorithms and DeepMind's Perceiver.

Literature Survey

Convolutional Neural Networks

COVID-X NET

It consists of five convolutional, four max pooling, a GAP, and a softmax layer[5]. Preprocessing steps were to convert the images into grayscale. Since the images were from different X-ray machines, a histogram matching process was applied to every image, taking one of them as a reference to equalize the histogram distribution. The rib cage shadows were suppressed using a pre-trained autoencoder model developed by Chuong M. Huynh, which makes it easier for the network to focus on relevant information within the lungs. COVID-XNet achieved a 93.14% F1 score and an AUC value of 0.988 on average over the different folds in 5 fold cross-validation.

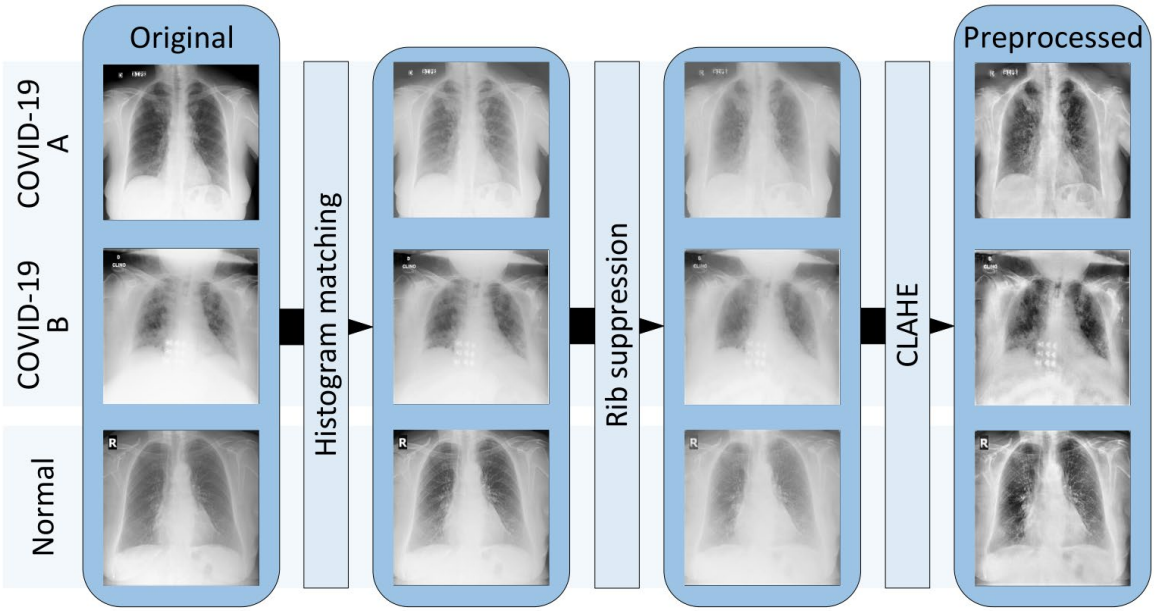


Fig. Image Preprocessing in Covid-X Net [5]

CoroNet

CoroNet architecture is based on Xception CNN architecture, 71 layers deep CNN architecture pre-trained on the ImageNet dataset[6]. Dataset source was from Github repository by Joseph et al with x-ray containing Normal, Pneumonia bacterial, Pneumonia viral images.

Performance of the CoroNet on Dataset-2

Class	Precision (%)	Recall (%)	Specificity (%)	F-measure (%)
COVID-19	97	89	99.6	93
Normal	92	85	97.7	89
Pneumonia Bacterial	87	95	88.7	91
Average	92	90	95.3	91
Overall Accuracy	90.21%			

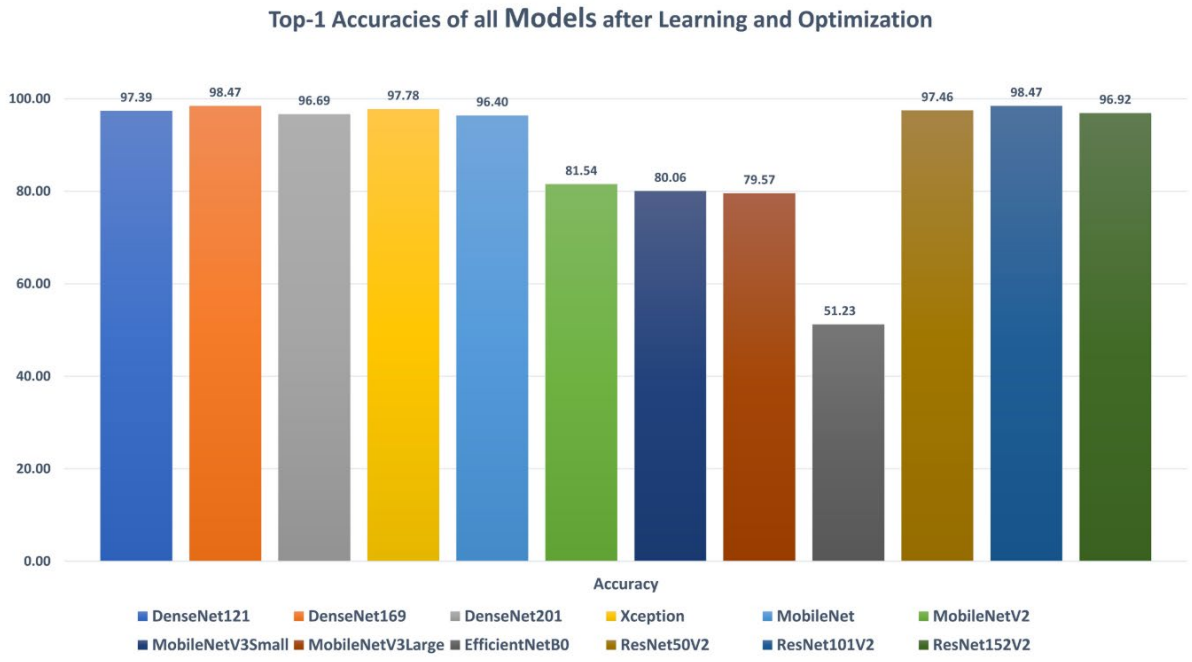
ELM Chimp

A two-phase approach for classifying chest X-ray images where the first phase is to use a deep CNN feature extractor, and the second phase using an Extreme Learning Machine, a single hidden layer neural network [7]. Chimp Optimization Algorithm was used to improve the performance and reliability of the network. The approach was benchmarked with 98.25 % and 99.11 % on the COVID-Xray-5k and COVIDetectioNet datasets, respectively. The major benefit was that about 0.9474 milliseconds was the training time,

and the total testing time was 2.937s for 3100 images. It consists of three convolutional, two pooling, a dense layer with 120 neurons and 2 neurons in the output layer.

Manta Ray Foraging Optimization

This paper presented an Optimized Transfer Learning approach with the Manta Ray Foraging Optimization algorithm to optimize the parameters of the following twelve available CNN architectures on chest X-ray image dataset which had been collected from eight public datasets containing 12,933 images [8]. Data augmentation was performed to increase the training set and enhance generalization. The results are shown below in the figure



Popular CNN Architectures

This study shows the performance of available CNN architectures such as VGG16, InceptionV3, ResNet50, DenseNet121 and Xception, and using their extracted features to predict using classic machine learning algorithms such as support vector machine, random forest, decision trees, bagging and AdaBoost classifiers [9]. 1102 balanced chest X-ray images in total were consolidated by combining three public datasets. The best performance was the accuracy of Xception paired with SVM at 99.33% which is 2.58% higher than stand-alone Xception. Individual CNN performances and best performance results are shown below.

Evaluation results of transfer learning methods.

Model	SEN%	SPE%	PRE%	ACC%	F1%	AUC%
VGG16 [19]	91.73 ± 0.34	98.97 ± 0.29	98.70 ± 0.36	95.64 ± 0.00	93.81 ± 1.82	95.34 ± 0.02
InceptionV3 [20]	92.46 ± 0.34	98.76 ± 0.00	98.45 ± 0.00	95.72 ± 0.17	95.36 ± 0.18	95.75 ± 0.12
ResNet50 [21]	89.29 ± 2.41	95.65± 0.51	94.59 ± 0.56	92.73 ± 1.03	91.85 ± 1.27	92.47 ± 1.14
DenseNet121 [22]	91.24 ± 0.60	98.35 ± 0.30	97.91 ± 0.37	95.08 ± 0.42	94.46 ± 0.48	94.75 ± 0.44
Xception [23]	**94.16 ± 0.60**	**99.17 ± 0.29**	**98.97 ± 0.36**	**96.75 ± 0.16**	**96.38 ± 0.18**	**96.54 ± 0.16**

Performance of the best method on other dataset

Method (best)	SEN%	SPE%	PRE%	ACC%	F1%	AUC%
InceptionV3 + SVM	78.38	99.33	96.67	95.19	86.57	94.38
InceptionV3 + Bagging	81.08	99.33	96.78	95.72	88.24	95.02
Xception + SVM	**81.08**	**99.33**	**96.78**	**95.72**	**88.24**	**95.02**
Xception + Bagging	81.08	99.33	96.78	95.27	88.24	95.02

The results from this study show that InceptionNetV3 and InceptionResNetV2 performed best on covid chest x-ray images with data augmentation[10]. Their results show performances of the 3 networks below. Other architecture results were also consolidated below from other researchers.

A comparison of training and validation accuracies of InceptionNetV3, InceptionResNetV2 and NASNetLarge.

-	Training Accuracy	Validation Accuracy	Data Augmentation
InceptionResNetV2	98.64%	97.87%	Yes
	99.47%	87.23%	No
InceptionNetV3	98.63%	97.87%	Yes
	98.90%	87.23%	No
NASNetLarge	99.54%	92%	Yes
	98.93%	61.70%	No

A comparison of different studies and models for detection of COVID-19.

Reference	Model	Accuracy
Narin *et al*. [13]	Inception-V3 ResNet50 Inception ResNet V2	100% 100% 95%
Wang *et al*. [16]	M-Inception	82.9%
Abbas *et al*. [30]	DeTrac	95.12%
Pathak *et al*. [18]	GCNN (proposed)	3.01%
Ardakani *et al*. [33]	AlexNet VGG-16 VGG-19 SqueezNet GoogleNet MobileNet-V2 ResNet-18 ResNet-50 ResNet-101 Xception	78.92% 83.33% 85.29% 82.84% 85.29% 92.16% 91.67% 94.12% 99.51% 99.02%

Xception

The Xception architecture is a linear stack of depth wise separable convolution layers with residual connections which outperforms Inception in ImageNet competition, and also has higher efficiency [21].

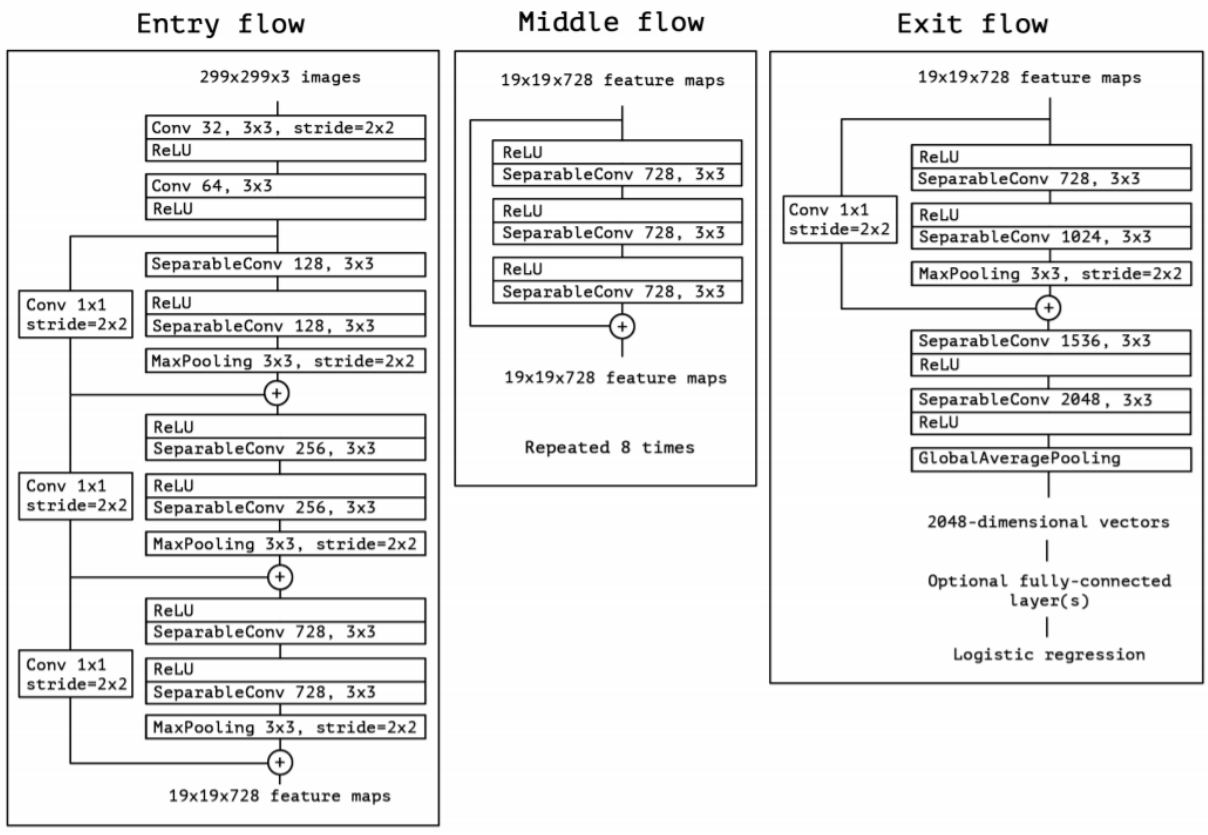


Fig. Xception Architecture [21]

VGG

VGG16 is a convolutional neural network with a depth of 16 layers that perform very well in several applications [22]. It was one of the very early architectures which performed very well in the ImageNet competition.

VGG19: is a variation of VGG16 where it contains 19 layers instead.

ResNet

A residual learning framework for training deeper networks with ease was developed. The layers were reformulated as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions [23]. These residual networks are easier to

optimize since the residual layers allow data from previous layers to permeate through the network, allowing easier training without vanishing gradients and gain accuracy from deeper networks.

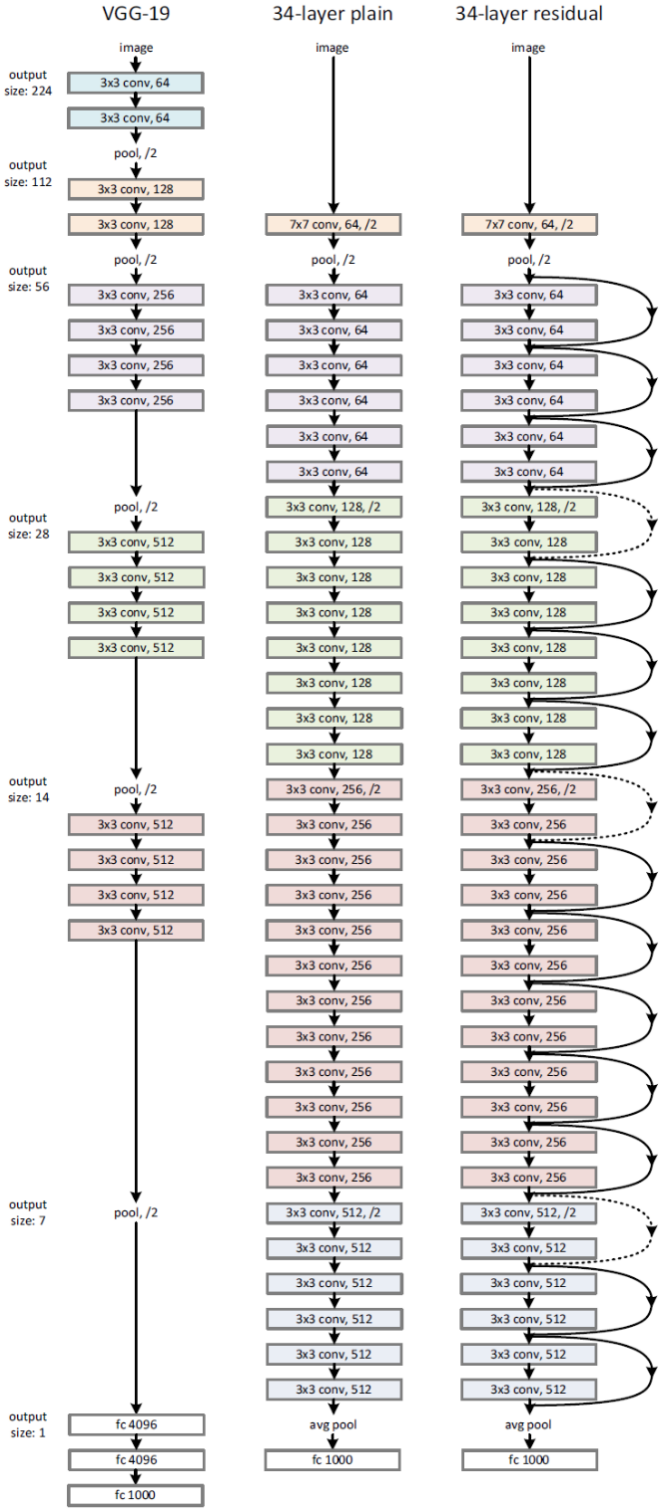


Fig. ResNet Architecture [23]

Resnet50, Resnet101, Resnet152 have 50, 101 and 152 layers respectively. The skip connections have been moved after the activation in V2.

DenseNet

They are densely connected convolutional networks where the feature maps from previous layers are concatenated with the inputs of the future layers [24]. This allows very deep networks to train without having the problem of vanishing gradients.

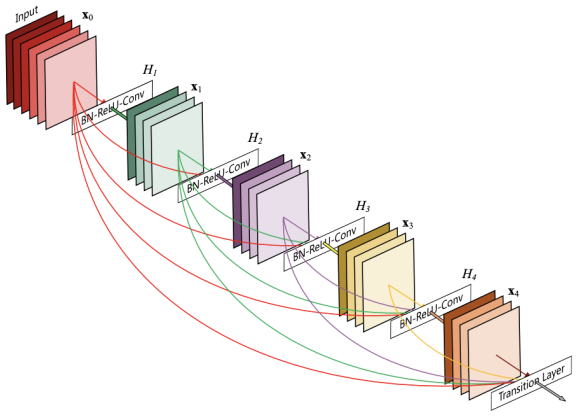


Fig. DenseNet Architecture [24]

DenseNet121, DenseNet169, DenseNet201 have 121, 169 and 201 layers respectively.

NASNet

Neural Architecture Search Net (NASNet) is a novel idea in which reinforcement learning is used to learn the architectures of cells within a network [25]. A controller RNN is used to predict the required architecture by making it pick from a defined set of layers followed by an element-wise addition or concatenation. The NASNet comes up with complex architectures with separate pathways for the data like in inception net, which tends to be specialised for the particular dataset.

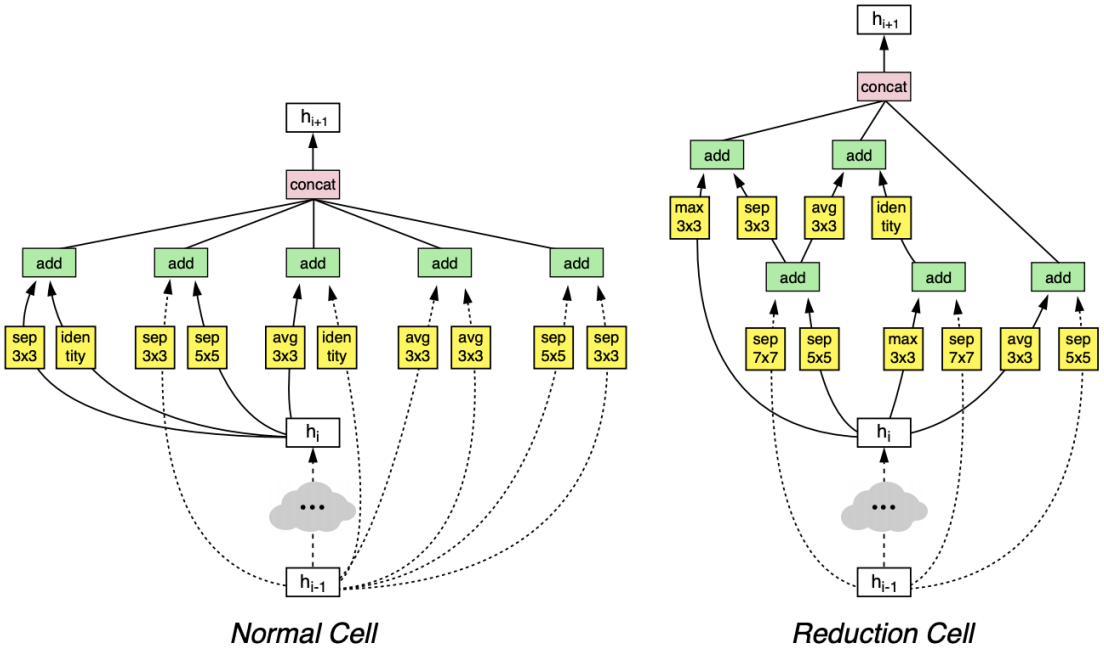
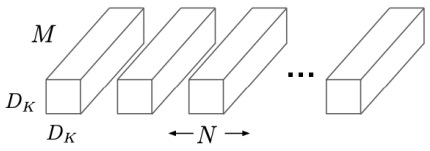


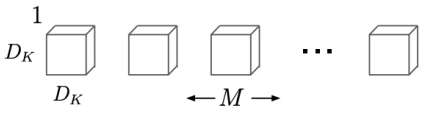
Fig. The architecture of the best convolutional cells (NASNet-A) [25]

MobileNet

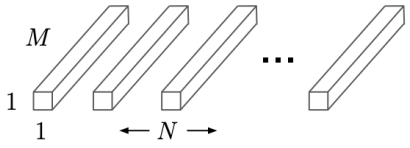
Depthwise separable convolution, that is the channels are separated depthwise and convoluted [26]. The above matrices are stacked. The stack is re-convoluted using a 1x1 filter. MobileNet and MobileNetV2 are used for better latency or accuracy respectively.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

InceptionResNet

It uses residual connection for inception blocks combing the advantages of both networks[27]

Genetic Algorithms

Genetic algorithm mimics the evolution of features by having a population of genes and using crossover between the best genes to produce offspring which are again added to the population[28]. A random mutation can be optionally included in order to create more variation.

Neuro-Evolution Augmenting Topologies

It is a genetic algorithm that generates artificial neural networks [29]. It is also capable of evolving the network topology over time. It does this by encoding the neural network architecture as a genome sequence. This genome sequence is used to generate the phenotype which is the actual neural network which is tested for the fitness. All crossover mutation is done in the genome. Hence, this enables it to change the architecture of the network over time.

Transformer Networks

Transformers have become an important and incredibly powerful part of Deep Learning. The idea of transformers has transformed the world of AI to be capable of unbelievably human-like performance in the fields of Natural Language Processing and Computer Vision. Big tech companies such as Google DeepMind and OpenAI have proven the power of transformers in Natural Language Processing using models such as BERT[30] and GPT[31].

“Attention Is All You Need”[32]. The title of the paper that is the cause of this revolution explains the whole intuition and concept behind transformers. Unlike what had been done historically in NLP and most of the deep learning, transformers seek to pay attention to only certain parts of the input at a time that is relevant to the task. Being capable of understanding the importance of certain parts of input highly boosts the ability of networks to truly understand the task rather than try to form an algebraic relation between inputs and outputs.

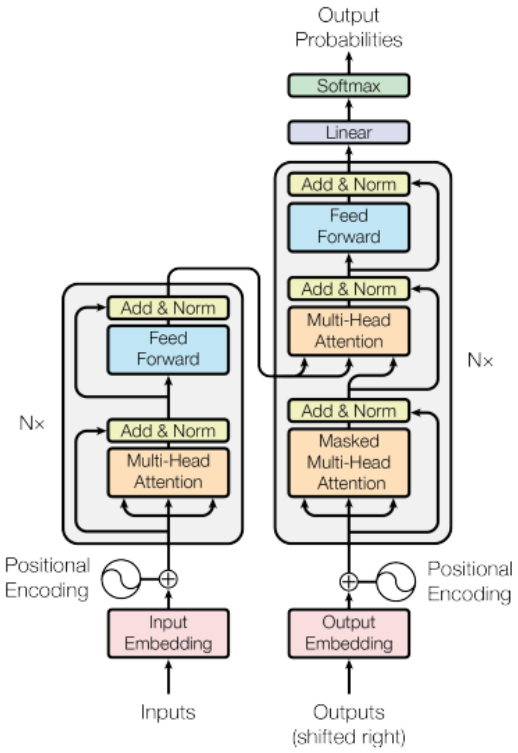


Fig. Transformer architecture [32]

Transformers follow an encoder-decoder architecture in which the encoder encodes the input data into an internal state that represents the input in a way understandable by the decoder which then uses this internal state to generate appropriate output.

The encoder works by producing a query-key-value vector triplet from a position-encoded input. This triplet concept can be better understood from the perspective of a database. The query vector, similar to a database query, is expected to be learnt to represent the overall expectation that the input contains. For example, for input of “How are you?”, we expect the query vector to represent the sentiment of asking about someone’s health and wellbeing. The key-value pair, similar to how they work in databases, is expected to be learnt in a form that using the key and an appropriate query, the expected value can be fetched or in this case, generated.

Matrix multiplication of the query and key vectors produces a square matrix that represents the importance of different parts of the input to each other. This concept is appropriately named self-attention since attention weights of different parts of the same input are being calculated.

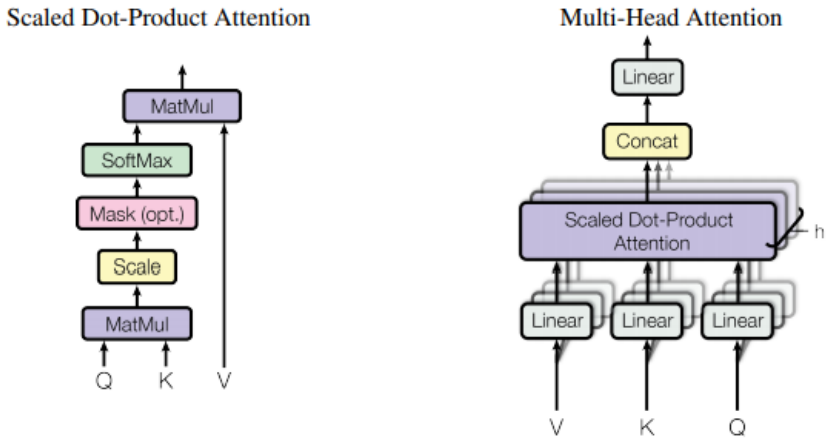


Fig. Multi-headed attention [32]

The final query and key vectors are passed to the decoder, and the decoder is expected to generate values that match the query-key pair. In NLP applications, the decoder is initially provided with a start token and in subsequent stages, fed back its own output as in the case of previous encoder-decoder networks in NLP.

Transformers require a lot of data compute power to train due to their unique architecture. Due to this, the training of transformers is not a feasible task for several independent and small-scale researchers.

Vision Transformer

The concept of transformer networks was later applied to images in Google's Brain Team's paper introducing the Vision Transformer (ViT)[33]. The vision transformer simply splits the image into several parts and feeds them as a sequence to the previously discussed transformer's encoder. After processing from the encoder, a simple dense layer is used for classification.

Despite being simple, this network has shown potential for significant improvements in the field of image classification.

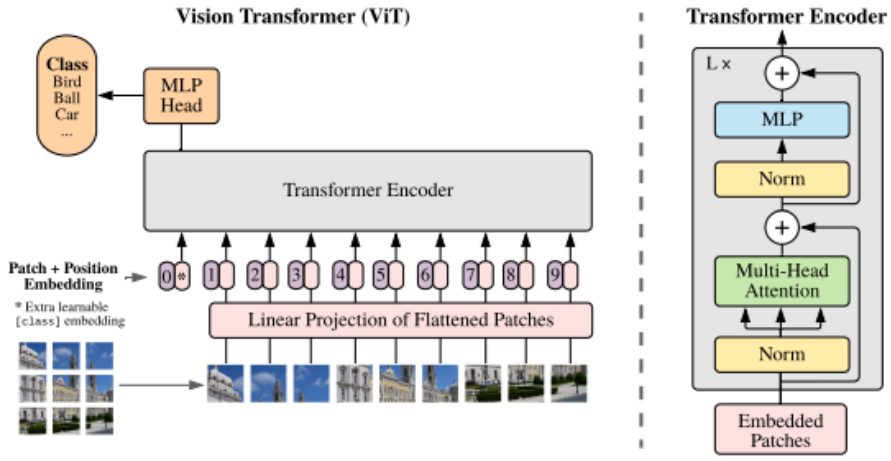


Fig. Vision Transformer (ViT) Architecture [33]

Perceiver

This brand new paper from Deepmind introducing the Perceiver network which has been published just a few months ago is a revolutionary breakthrough in artificial intelligence and deep learning[34]. Up till now, neural networks were specialized to certain modalities of data; for example, CNNs are good for image data and RNNs are good for sequences. The Perceiver is the first neural network capable of handling multiple modalities of data. According to the paper, the network is capable of handling image, video, audio and point-cloud data. This capability of handling several modalities is a big step towards artificial general intelligence.

While the network itself hasn't shown competency against existing methods for certain tasks, it is important to recognise that this is the first paper introducing such an innovative architecture and with time, it is only going to be improved.

The way the perceiver handles such varied modalities of data is really interesting. It uses cross-attention modules to convert the high-dimensional input data to a fixed-size latent space that can be understood and further processed by the network.

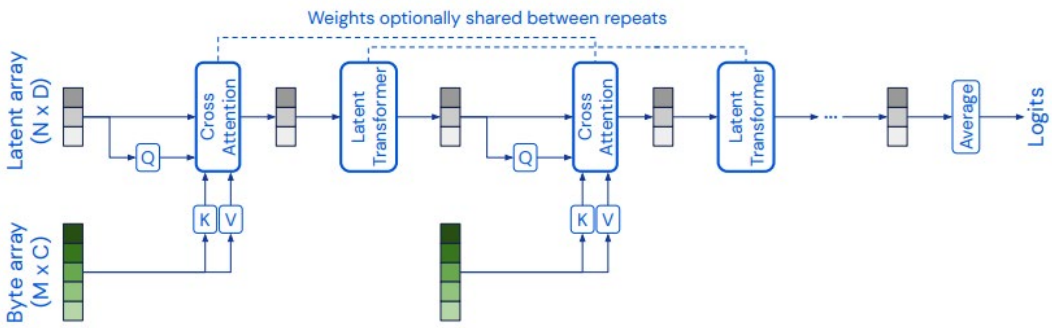


Fig. Perceiver architecture [34]

Despite having been released just months back, the Perceiver already has several implementations on GitHub with popular frameworks such as TensorFlow, Keras and PyTorch. The community is definitely excited and anticipating more and better versions of this network.

Methodology

Problem Definition

The goal of this research is to try out new techniques in deep learning to identify covid from chest x rays and evaluate them.

Proposed Approach

To initially train and evaluate existing CNN architectures to obtain a baseline of their performance in raw downscaled image data. From the trained networks, we will select 5 best performing distinct networks and try different ensembling techniques to evaluate.
We will evaluate the difference of using genetic algorithms to optimize the weights of a network against gradient based methods.
We will also try out the Neuro-Evolution of Augmenting Topologies (NEAT) algorithm to find a more efficient and optimized network architecture.
Finally we will try training the new Deepmind perceiver architecture and check its performance.

Dataset

The dataset used in this research is the COVID-19 RADIOGRAPHY DATABASE from Kaggle [35 - 36]. It consists of 3616 COVID-19 positive cases along with 10,192 Normal, 6012 Lung Opacity (Non-COVID lung infection), and 1345 Viral Pneumonia images. Only the covid and normal x rays were used for this research. These images are grayscale and of size 299x299.

Implementation

Image pre-processing

There were 10192 normal chest x ray and 3616 covid chest x ray images which were read, converted from 299x299 into a 128x128 3 channel array (using Cubic downscaling technique) since most of the deeper CNNs work only with 3 channel images and the image resolution was reduced for faster computation due to limited computational resources. The images weren't directly normalized since the float value made the files more than 30GB which overloads my computer's RAM. Therefore, they were stored as uint8 format in a numpy file totaling around 700MB. A normalization layer from tensorflow-keras-experimental was used after the input layer in all of the CNNs, although they were later removed since they caused problems hindering the optimizer and it wasn't actually required. A balanced data set was also created and when tested, the models learnt well with the unbalanced dataset, hence this was discarded as well.

Baseline CNN

In order to find out the performance of CNNs without any histogram equalization, data balancing, data augmentation since the Xray images had the same specification and were purely trained on raw data. The networks Xception, VGG16, VGG19, ResNet50, ResNet101, ResNet152, ResNet50V2, ResNet101V2, InceptionV3, InceptionResNetV2, MobileNet, MobileNetV2, DenseNet121, DenseNet169, DenseNet201, NASNetMobile, NASNetLarge were trained.

Hardware and framework

Initially the NVIDIA GTX 1060 MAX-Q GPU in my laptop was used to train the models but it took way too long. Hence, NVIDIA TESLA V100 x1 GPU (from CIT Compute Cluster) was used for the training of the CNNs which sped up the process by a factor of 4. The networks were trained using the Keras framework in TensorFlow.

Training

The models were trained using Stochastic gradient descent optimizer with a batch size of 16. Early stopping callback with patience=10 based on accuracy was used which will restore the best weights at the end. All architectures were trained from scratch and no previously trained weights such as imagenet weights were used.

Followed by the convolutional layers, 2 dense layers of 256 neurons each with a dropout of 0.5 and relu activation followed by a single sigmoid neuron for the output.

Ensemble

Out of the above baseline models, VGG16, ResNet50V2, Xception, ResNet101V2, DenseNet169 5 were selected for their highest performance on training and validation accuracy with the least training time.

Mean Ensemble

The 5 selected trained models were made to predict individually and their sigmoid outputs were averaged as an ensemble which outperformed the individual learners.

ELM Ensemble

Following the previous experiment, based on individual learner predictions, their outputs were given as inputs to a single hidden layer neural network with 32 neurons and trained as an ensemble.

Deeper ANN Ensemble

Based on individual learner predictions, their outputs were given as inputs to a deeper neural network and trained as an ensemble.

The ensemble networks were trained on the NVIDIA GTX 1060 MAX-Q GPU using TensorFlow Keras with Adam optimizer and had early stopping callback patience as 50.

Genetic algorithm for optimization

KerasGA [37] python library (TensorFlow-Keras based) was used to try to optimize the weights of a shallow CNN network containing 2 convolution layers, a pooling layer and a dense layer followed by the output layer using the NVIDIA GTX 1060 MAX-Q GPU.

NEAT algorithm

A network topology was tried to be created and weights were tried to be optimized by a NEAT algorithm using PyTorch-NEAT library in the NVIDIA GTX 1060 MAX-Q GPU.

Perceiver

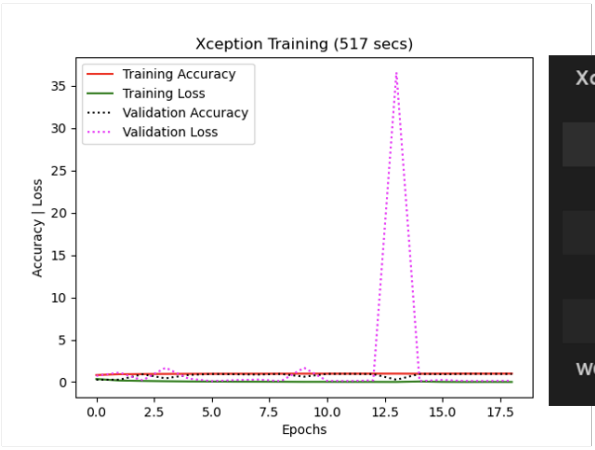
The original code from DeepMind was too complex to work with given the timeframe and required knowledge of unfamiliar frameworks. Later, Keras io for perceiver [38] was found online but it had too many errors that would pop up one by one when each one of them was fixed due to unnecessary components. Finally, a PyTorch wrapper [39] was found, and it was used to train the perceiver.

In the beginning, it was trained using the NVIDIA GTX 1060 MAX-Q GPU where an epoch would train for around half an hour. Different parameters such as learning rate were tweaked and finalized to 0.001 with Stochastic Gradient Descent Optimizer. After trying out different settings, the best setting was chosen. More than 50 epochs later (around 20 hours later), the training history revealed that there was a constant decrease in loss but it was not close enough for deployment. Therefore, after trying with major cloud providers (since CIT revoked the GPU cluster access since it was the end of the academic year) such as Google Cloud Platform, Amazon web services, Microsoft Azure who refused to grant manual access to their GPU within time, a TESLA A100 GPU was leased online on an hourly basis in e2enetworks.com, a local provider who immediately granted access and the training process began. A limited number of epochs was trained within the available budget.

Results

Convolutional Neural Networks

Xception

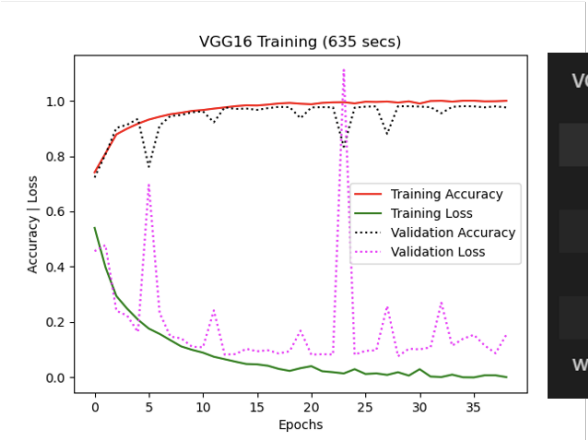


Xception

	precision	recall	f1-score	support
0	0.929412	0.928198	0.928805	766.000000
1	0.972459	0.972946	0.972702	1996.000000
accuracy	0.960536	0.960536	0.960536	0.960536
macro avg	0.950935	0.950572	0.950753	2762.000000
weighted avg	0.960520	0.960536	0.960528	2762.000000

Training Accuracy: 0.9967, Validation Accuracy: 0.9576, Epochs:19
This model has a good fit and performs well with minimal epochs

VGG16

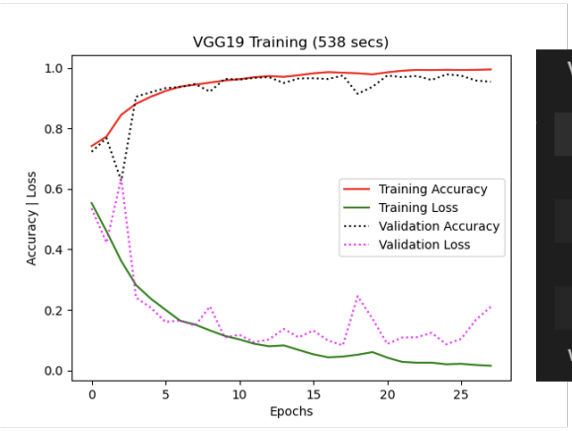


VGG16

	precision	recall	f1-score	support
0	0.963494	0.964752	0.964123	766.000000
1	0.986466	0.985972	0.986219	1996.000000
accuracy	0.980087	0.980087	0.980087	0.980087
macro avg	0.974980	0.975362	0.975171	2762.000000
weighted avg	0.980095	0.980087	0.980091	2762.000000

Training Accuracy: 0.9997, Validation Accuracy: 0.9757, Epochs: 40
This network had a good fit until it started to overfit in the last few epochs.

VGG19

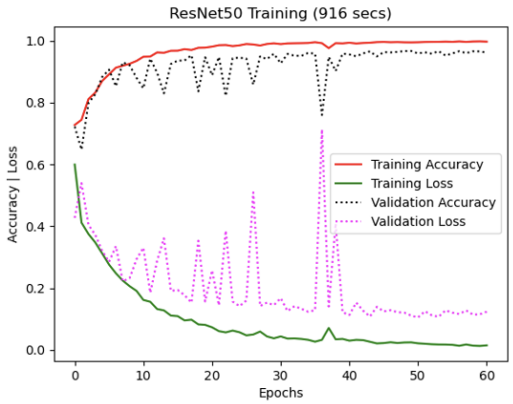


VGG19

	precision	recall	f1-score	support
0	0.970270	0.937337	0.953519	766.000000
1	0.976261	0.988978	0.982578	1996.000000
accuracy	0.974656	0.974656	0.974656	0.974656
macro avg	0.973266	0.963157	0.968049	2762.000000
weighted avg	0.974600	0.974656	0.974519	2762.000000

Training Accuracy: 0.9944, Validation Accuracy: 0.9540, Epochs: 28
This network had a good fit until it started to overfit in the last few epochs.

ResNet50

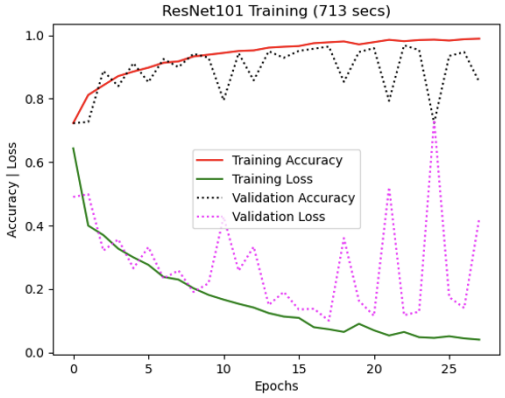


ResNet50

	precision	recall	f1-score	support
0	0.901720	0.958225	0.929114	766.00000
1	0.983573	0.959920	0.971602	1996.00000
accuracy	0.959450	0.959450	0.959450	0.95945
macro avg	0.942646	0.959072	0.950358	2762.00000
weighted avg	0.960872	0.959450	0.959819	2762.00000

Training Accuracy: 0.9972, Validation Accuracy: 0.9619, Epochs: 60
The network trained well with a decent fit (very mild overfitting observed). It required more epochs to train.

ResNet101

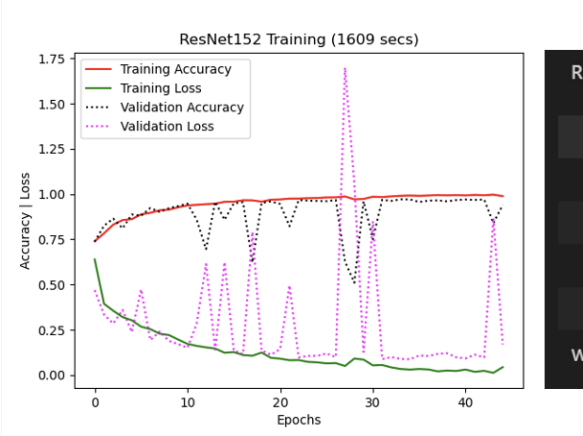


ResNet101

	precision	recall	f1-score	support
0	0.941722	0.928198	0.934911	766.000000
1	0.972596	0.977956	0.975269	1996.000000
accuracy	0.964156	0.964156	0.964156	0.964156
macro avg	0.957159	0.953077	0.955090	2762.000000
weighted avg	0.964033	0.964156	0.964076	2762.000000

Training Accuracy: 0.9890, Validation Accuracy: 0.8526, Epochs: 29
The network trained unstably with heavy overfits with lesser number of epochs.

ResNet152

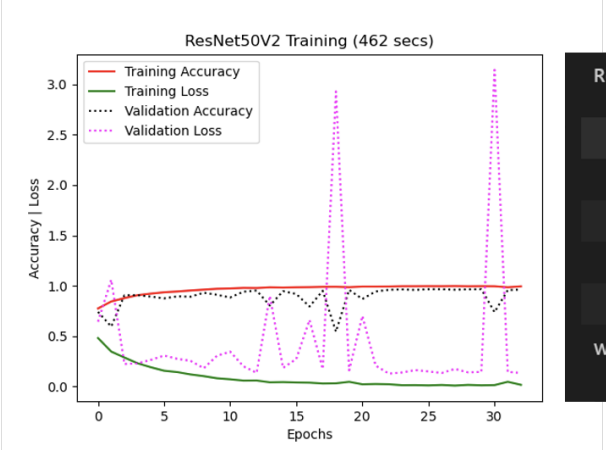


ResNet152

	precision	recall	f1-score	support
0	0.932911	0.962141	0.947301	766.000000
1	0.985294	0.973447	0.979335	1996.000000
accuracy	0.970311	0.970311	0.970311	0.970311
macro avg	0.959103	0.967794	0.963318	2762.000000
weighted avg	0.970767	0.970311	0.970451	2762.000000

Training Accuracy: 0.9884, Validation Accuracy: 0.9402, Epochs: 45
The network trained well with good fit even though it needed more number of epochs.

ResNet50V2

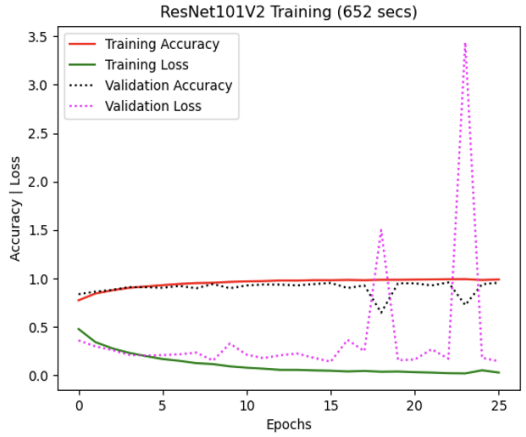


ResNet50V2

	precision	recall	f1-score	support
0	0.942049	0.912533	0.927056	766.000000
1	0.966832	0.978457	0.972610	1996.000000
accuracy	0.960174	0.960174	0.960174	0.960174
macro avg	0.954440	0.945495	0.949833	2762.000000
weighted avg	0.959958	0.960174	0.959976	2762.000000

Training Accuracy: 0.9944, Validation Accuracy: 0.9623, Epochs: 33
The network trained very well with good fit even with a medium number of epochs.

ResNet101V2

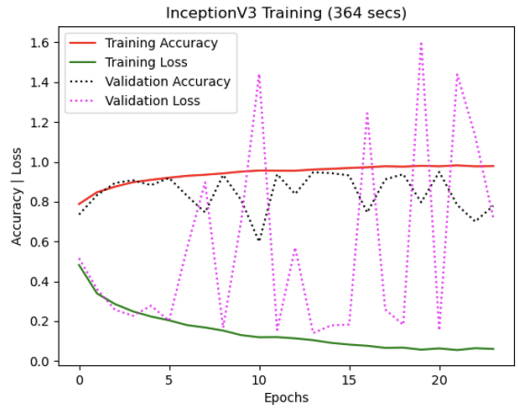


ResNet101V2

	precision	recall	f1-score	support
0	0.926667	0.907311	0.916887	766.000000
1	0.964712	0.972445	0.968563	1996.000000
accuracy	0.954381	0.954381	0.954381	0.954381
macro avg	0.945689	0.939878	0.942725	2762.000000
weighted avg	0.954160	0.954381	0.954231	2762.000000

Training Accuracy: 0.9900, Validation Accuracy: 0.9572, Epochs: 25
The network trained excellently with mild overfitting but has a low number of epochs.

InceptionV3

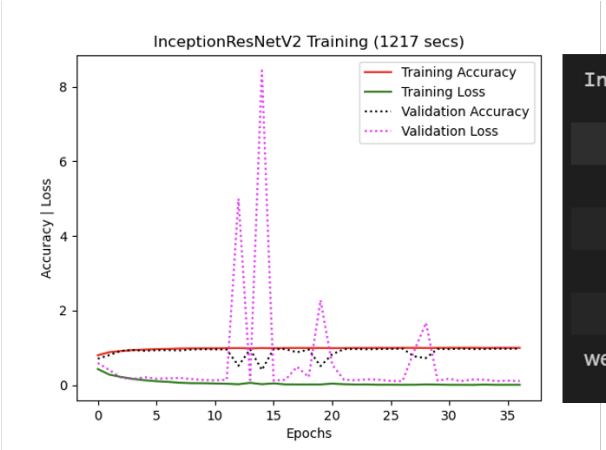


InceptionV3

	precision	recall	f1-score	support
0	0.926027	0.882507	0.903743	766.000000
1	0.955709	0.972946	0.964250	1996.000000
accuracy	0.947864	0.947864	0.947864	0.947864
macro avg	0.940868	0.927726	0.933997	2762.000000
weighted avg	0.947477	0.947864	0.947470	2762.000000

Training Accuracy: 0.9785, Validation Accuracy: 0.7791, Epochs: 24
The network has an unstable training history with heavy overfitting.

InceptionResNetV2

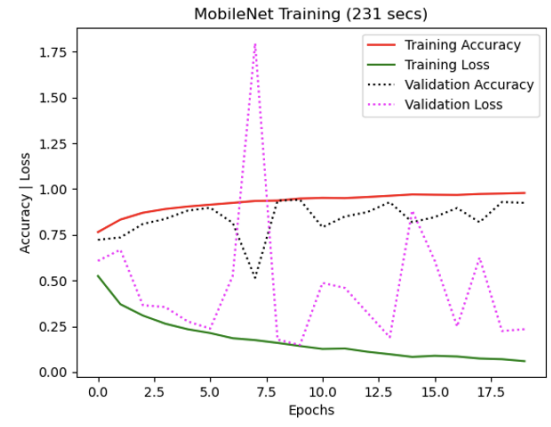


InceptionResNetV2

	precision	recall	f1-score	support
0	0.973009	0.941253	0.956868	766.000000
1	0.977734	0.989980	0.983819	1996.000000
accuracy	0.976466	0.976466	0.976466	0.976466
macro avg	0.975372	0.965617	0.970343	2762.000000
weighted avg	0.976424	0.976466	0.976344	2762.000000

Training Accuracy: 0.9983, Validation Accuracy: 0.9753, Epochs: 36
The network has trained excellently with a medium number of epochs.

MobileNet

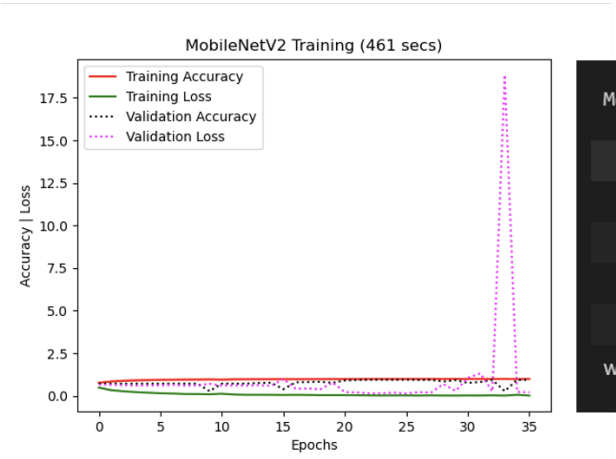


MobileNet

	precision	recall	f1-score	support
0	0.924157	0.859008	0.890392	766.000000
1	0.947317	0.972946	0.959960	1996.000000
accuracy	0.941347	0.941347	0.941347	0.941347
macro avg	0.935737	0.915977	0.925176	2762.000000
weighted avg	0.940894	0.941347	0.940667	2762.000000

Training Accuracy: 0.9786, Validation Accuracy: 0.9246, Epochs: 19
The network has a medium overfitting character but has very low number of epochs

MobileNetV2

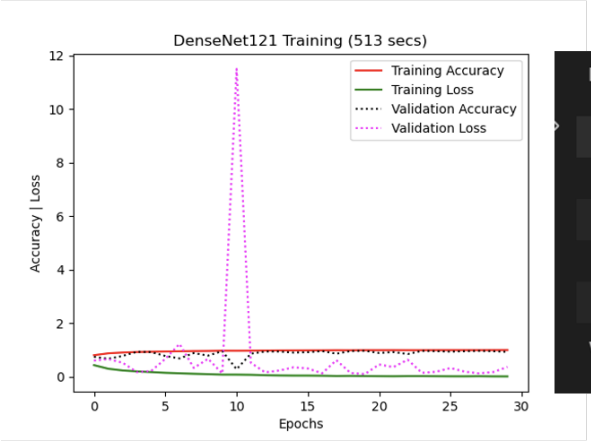


MobileNetV2

	precision	recall	f1-score	support
0	0.935047	0.902089	0.918272	766.000000
1	0.962926	0.975952	0.969395	1996.000000
accuracy	0.955467	0.955467	0.955467	0.955467
macro avg	0.948987	0.939020	0.943834	2762.000000
weighted avg	0.955195	0.955467	0.955217	2762.000000

Training Accuracy: 0.9945, Validation Accuracy: 0.9456, Epochs: 35
The network has trained well with medium overfitting.

DenseNet121

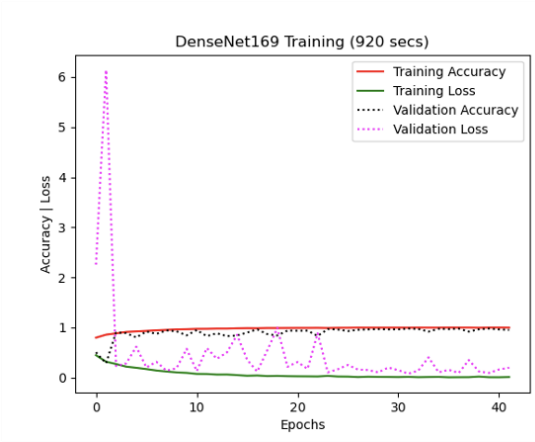


DenseNet121

	precision	recall	f1-score	support
0	0.953209	0.930809	0.941876	766.000000
1	0.973684	0.982465	0.978055	1996.000000
accuracy	0.968139	0.968139	0.968139	0.968139
macro avg	0.963446	0.956637	0.959965	2762.000000
weighted avg	0.968006	0.968139	0.968021	2762.000000

Training Accuracy: 0.9969, Validation Accuracy: 0.9283, Epochs: 29
The network has trained well with moderate overfitting.

DenseNet169

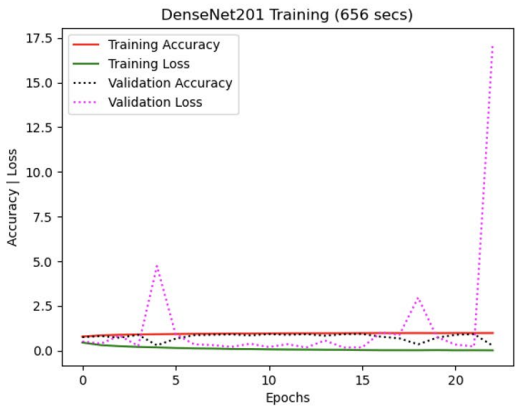


DenseNet169

	precision	recall	f1-score	support
0	0.951282	0.968668	0.959897	766.000000
1	0.987891	0.980962	0.984414	1996.000000
accuracy	0.977552	0.977552	0.977552	0.977552
macro avg	0.969587	0.974815	0.972155	2762.000000
weighted avg	0.977738	0.977552	0.977615	2762.000000

Training Accuracy: 0.9963, Validation Accuracy: 0.9507, Epochs: 42
This network has a slight overfit with a medium number of epochs.

DenseNet201

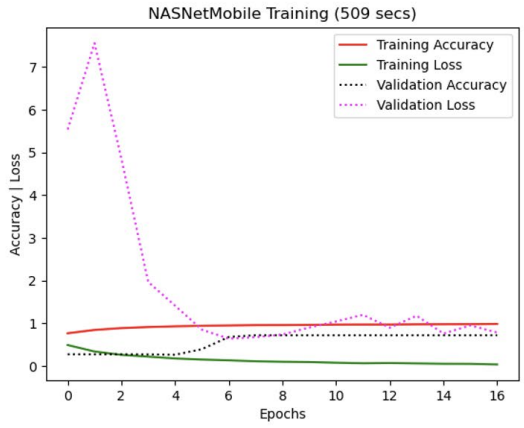


DenseNet201

	precision	recall	f1-score	support
0	0.800000	0.981723	0.881594	766.000000
1	0.992316	0.905812	0.947093	1996.000000
accuracy	0.926865	0.926865	0.926865	0.926865
macro avg	0.896158	0.943767	0.914344	2762.000000
weighted avg	0.938980	0.926865	0.928928	2762.000000

Training Accuracy: 0.9963, Validation Accuracy: 0.9507, Epochs: 24
This network has a slight overfit with a low number of epochs.

NASNetMobile

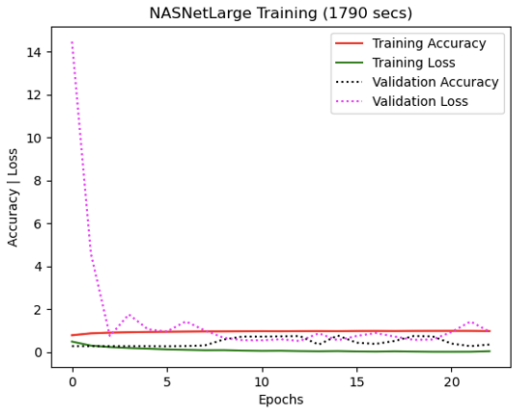


NASNetMobile

	precision	recall	f1-score	support
0	0.316294	0.129243	0.183503	766.000000
1	0.727644	0.892786	0.801800	1996.000000
accuracy	0.681028	0.681028	0.681028	0.681028
macro avg	0.521969	0.511014	0.492652	2762.000000
weighted avg	0.613562	0.681028	0.630324	2762.000000

Training Accuracy: 0.9866, Validation Accuracy: 0.7226, Epochs: 16
This network is highly overfitted and has a very low number of epochs.

NASNetLarge



NASNetLarge

	precision	recall	f1-score	support
0	0.603933	0.280679	0.383244	766.000000
1	0.770989	0.929359	0.842799	1996.000000
accuracy	0.749457	0.749457	0.749457	0.749457
macro avg	0.687461	0.605019	0.613021	2762.000000
weighted avg	0.724659	0.749457	0.715348	2762.000000

Training Accuracy: 0.9851, Validation Accuracy: 0.3533, Epochs: 23
This network is highly overfitted.

From the above baseline models, the results were sorted based on performance on training and validation accuracy with the least training time as shown in the table below.

	File_name	Net_name	Training_accuracy	Validation_accuracy	Elapsed_time
1	b2	VGG16	0.999728	0.975742	634.559963
9	b11	InceptionResNetV2	0.998370	0.975380	1217.068339
6	b7	ResNet50V2	0.994478	0.962346	462.197762
3	b4	ResNet50	0.997284	0.961984	915.696494
0	b1	Xception	0.996741	0.957639	517.360124
7	b8	ResNet101V2	0.990042	0.957277	651.653594
2	b3	VGG19	0.994478	0.954019	538.046155
13	b15	DenseNet169	0.996379	0.950760	919.578856
11	b13	MobileNetV2	0.994568	0.945692	461.010062
5	b6	ResNet152	0.988412	0.940261	1609.117441
12	b14	DenseNet121	0.996922	0.928313	513.084586
10	b12	MobileNet	0.978635	0.924692	231.469443
4	b5	ResNet101	0.989046	0.852643	712.883352
8	b10	InceptionV3	0.978544	0.779146	363.898605
15	b17	NASNetMobile	0.986692	0.722665	509.120492
16	b18	NASNetLarge	0.985153	0.353367	1790.174225
14	b16	DenseNet201	0.991490	0.282766	656.244360

Therefore, VGG16, ResNet50V2, Xception, ResNet101V2, DenseNet169 5 were selected in a manner to have diverse architectures.

Ensemble

Mean Ensemble

The selected trained models were made to predict individually and their sigmoid outputs were averaged as an ensemble. The results show that this technique outperforms the individual learners. The tables below show the performance on the training data and the validation data respectively.

Training Accuracy Ensemble 1

	precision	recall	f1-score	support
0	0.998948	1.000000	0.999474	2850.000000
1	1.000000	0.999634	0.999817	8196.000000
accuracy	0.999728	0.999728	0.999728	0.999728
macro avg	0.999474	0.999817	0.999645	11046.000000
weighted avg	0.999729	0.999728	0.999728	11046.000000

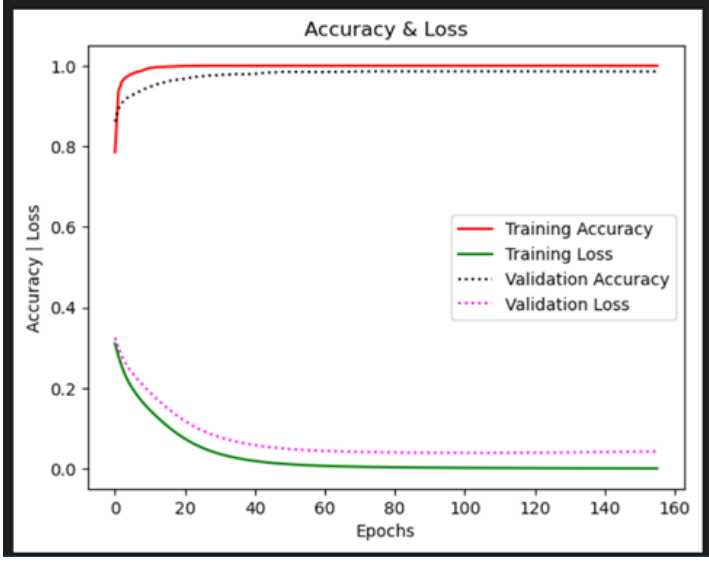
Validation Accuracy Ensemble 1

	precision	recall	f1-score	support
0	0.977661	0.971279	0.974460	766.00000
1	0.989005	0.991483	0.990243	1996.00000
accuracy	0.985880	0.985880	0.985880	0.98588
macro avg	0.983333	0.981381	0.982351	2762.00000
weighted avg	0.985859	0.985880	0.985866	2762.00000

ELM Ensemble

This is a network containing 32 neuron hidden layer and a single neuron output layer. It was trained based on the sigmoid predictions of the above selected networks and tested as shown below.

```
11/11 [==============================] - 0s 7ms/step - loss: 3.7611e-04 - accuracy: 1.0000 - val_loss: 0.0426 - val_accuracy: 0.9859
12.4546959400177 secs
<Figure size 640x480 with 0 Axes>
```



Ensemble 2

	precision	recall	f1-score	support
0	0.980106	0.964752	0.972368	766.000000
1	0.986554	0.992485	0.989510	1996.000000
accuracy	0.984794	0.984794	0.984794	0.984794
macro avg	0.983330	0.978618	0.980939	2762.000000
weighted avg	0.984766	0.984794	0.984756	2762.000000

This network performs well, although Mean ensemble outperforms it.

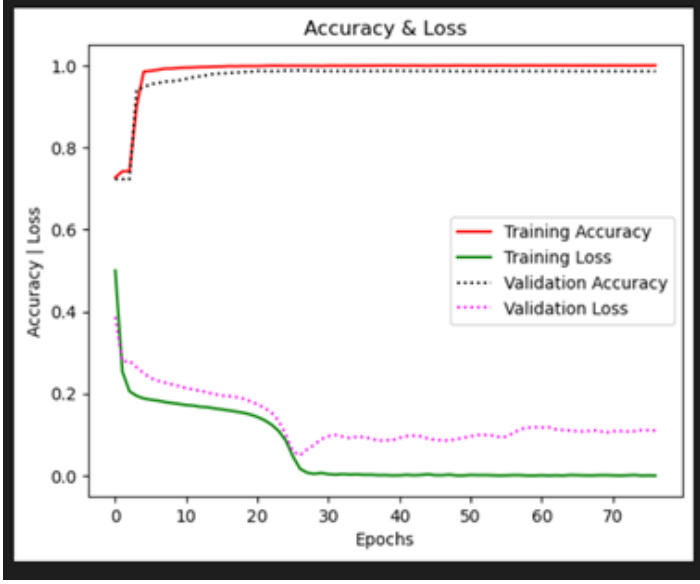
Deeper ANN Ensemble

This network containing the architecture shown below was trained based on the sigmoid predictions of the above selected networks and tested.

Layer (type)	Output Shape	Param #
dense_25 (Dense)	(None, 32)	192
dropout_6 (Dropout)	(None, 32)	0
dense_26 (Dense)	(None, 64)	2112
dropout_7 (Dropout)	(None, 64)	0
dense_27 (Dense)	(None, 128)	8320
dropout_8 (Dropout)	(None, 128)	0
dense_28 (Dense)	(None, 64)	8256
dense_29 (Dense)	(None, 32)	2080
dense_30 (Dense)	(None, 16)	528
dense_31 (Dense)	(None, 1)	17

Total params: 21,505
Trainable params: 21,505
Non-trainable params: 0

```
11/11 [==============================] - 0s 8ms/step - loss: 3.2666e-04 - accuracy: 1.0000 - val_loss: 0.1101 - val_accuracy: 0.9859
8.50852084159851 secs
```



Ensemble 3

	precision	recall	f1-score	support
0	0.977836	0.979112	0.978474	766.000000
1	0.991980	0.991483	0.991731	1996.000000
accuracy	0.988052	0.988052	0.988052	0.988052
macro avg	0.984908	0.985298	0.985102	2762.000000
weighted avg	0.988057	0.988052	0.988055	2762.000000

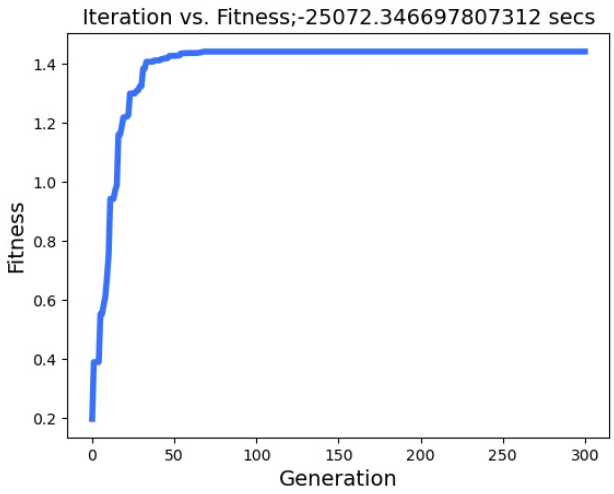
This network outperforms both of the above 2 ensemble networks.

Genetic Algorithm for Optimization

A shallow CNN network model was defined as shown below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 128, 128, 3)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 122, 122, 5)       740
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 5)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 22, 22, 3)         138
_________________________________________________________________
flatten (Flatten)            (None, 1452)              0
_________________________________________________________________
dense (Dense)                (None, 15)                21795
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 16
=================================================================
Total params: 22,689
Trainable params: 22,689
Non-trainable params: 0
```

The task of the genetic algorithm was to optimize the weights of the network to learn the data. The fitness is the inverse of Binary cross entropy loss. From the graph below we can see that even with 300 generations, it is incapable of optimizing the network due to the large number of parameters.



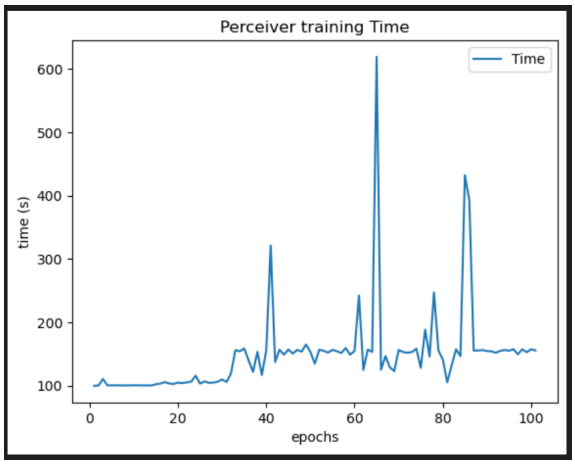
Neuro-Evolution for Augmenting Topologies

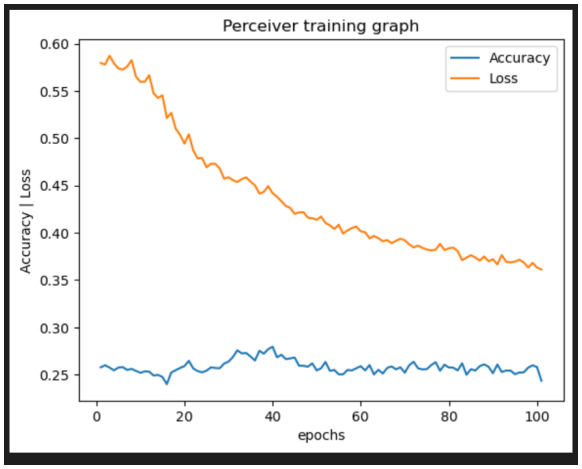
The algorithm was applied on 128x128 images. This time, the fitness was binary cross entropy and was to be minimized. Even after several hours, not even the first generation was completed. Therefore, in order to check if the code was correct and if the network worked, the images were down sampled to 10x10 and the code was rerun. Unsurprisingly the code worked and the algorithm tried to optimize it, but since there were too little

features that were non correlated, there was nothing for the network to learn and hence the fitness was oscillating,

Perceiver

From the training graph below, we can see that the network is learning steadily for the 100 epochs. A higher learning rate in the optimizer resulted in fluctuation of no proper learning of the network. Lower learning rate also had a steady drop in the loss, but it rate was too slow.





Perceiver training time:15006 secs

Conclusion

Convolutional Neural Networks

The baseline CNN algorithms performance is not a major highlight of this research, yet the results correlate with other research papers who have performed different techniques using the CNN architectures.

Ensemble

The mean ensemble which uses the mean of the individual learner predictions performed very well surpassing the individual learners.

ELM ensemble performs well, although mean ensemble performs better.

Deeper ANN ensemble performs better than both of the previous ensembles due to its ability to learn and store information of higher complexity.

These networks perform competitively with other networks where the authors have used 299x299 images whereas ours used 128x128 which reduces significant computation.

The reason for ensemble networks performing better could be due to the nature of working of the individual learners. They have vastly different architectures, different depths, different advantages and disadvantages. For example, some might deal with edge cases better, others might perform well on the majority of the data. The idea with the ensemble network, and the final neural network is to understand and avail the strengths of the individual learners and shadow their weaknesses which seems to work, resulting in higher accuracy.

Genetic Algorithm

The genetic algorithm fails to optimize the vast number of parameters of existing architecture even with hundreds of iterations. Therefore, the genetic algorithm is not suitable for updating the existing weights and falls incomparable to gradient descent based algorithms.

NEAT algorithm

The failure to update an existing architecture's weights sparked the idea of creating a network as simple as possible with maximum performance so that there would be a minimum number of parameters to optimize. However, this required too much computation to even get the first generation of population. I tried to optimize a very small dimension of the image (10x10) to check if the network worked, and it did, but too small an image has no information for the network to learn and is not viable. Therefore, the NEAT algorithm can be deemed incompatible for this application due to its extreme computational requirements.

Perceiver

This network started learning steadily with the right settings and parameters, but is a slow learner, since even at 100 epochs there is no significant result (higher learning rate didn't reduce the number of epochs, but the loss was oscillating). We can infer that this network is capable of learning for this application, however it requires very high computational power.

Bibliography

[1] "Who covid-19 situation reports," 2021. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus2019/situation-reports

[2] Dina M. Ibrahim, Nada M. Elshennawy, Amany M. Sarhan, Deep-chest: Multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases, Computers in Biology and Medicine, Volume 132, 2021, 104348, ISSN 0010-4825, https://doi.org/10.1016/j.compbiomed.2021.104348.

[3] Horry, Mike & Chakraborty, Subrata & Paul, Manoranjan & Ulhaq, Anwaar & Pradhan, Biswajeet & Saha, Manas & Shukla, Nagesh. (2020). COVID-19 Detection through Transfer Learning using Multimodal Imaging Data. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3016780.

[4] Rousan, L.A., Elobeid, E., Karrar, M. et al. Chest x-ray findings and temporal lung changes in patients with COVID-19 pneumonia. BMC Pulm Med 20, 245 (2020). https://doi.org/10.1186/s12890-020-01286-5

[5] Duran-Lopez, L., Dominguez-Morales, J. P., Corral-Jaime, J., Vicente-Diaz, S., & Linares-Barranco, A. (2020). COVID-XNet: A custom deep learning system to diagnose and locate COVID-19 in chest x-ray images. Applied Sciences (Switzerland). https://doi.org/10.3390/app10165683

[6] Khan, A. I., Shah, J. L., & Bhat, M. M. (2020). CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images. Computer Methods and Programs in Biomedicine. https://doi.org/10.1016/j.cmpb.2020.105581

[7] Hu, T., Khishe, M., Mohammadi, M., Parvizi, G. R., Taher Karim, S. H., & Rashid, T. A. (2021). Real-time COVID-19 diagnosis from X-Ray images using deep CNN and extreme learning machines stabilized by chimp optimization algorithm. Biomedical Signal Processing and Control. https://doi.org/10.1016/j.bspc.2021.102764

[8] Bahgat, W. M., Balaha, H. M., AbdulAzeem, Y., & Badawy, M. M. (2021). An Optimized Transfer Learning-based Approach for Automatic Diagnosis of COVID-19 from Chest X-ray Images. PeerJ Computer Science. https://doi.org/10.7717/PEERJ-CS.555

[9] Wang, D., Mo, J., Zhou, G., Xu, L., & Liu, Y. (2020). An efficient mixture of deep and machine learning models for COVID-19 diagnosis in chest X-ray images. PLoS ONE. https://doi.org/10.1371/journal.pone.0242535

[10] Albahli, S., & Albattah, W. (2020). Deep Transfer Learning for COVID-19 Prediction: Case Study for Limited Data Problems. Current Medical Imaging Formerly Current Medical Imaging Reviews. https://doi.org/10.2174/1573405616666201123120417

[11] Narayan Das, N., Kumar, N., Kaur, M., Kumar, V., & Singh, D. (2020). Automated Deep Transfer Learning-Based Approach for Detection of COVID-19 Infection in Chest X-rays. IRBM. https://doi.org/10.1016/j.irbm.2020.07.001

[12] Abbas, A., Abdelsamea, M. M., & Gaber, M. M. (2021). Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network. Applied Intelligence. https://doi.org/10.1007/s10489-020-01829-7

[13] Abraham, B., & Nair, M. S. (2020). Computer-aided detection of COVID-19 from X-ray images using multi-CNN and Bayesnet classifier. Biocybernetics and Biomedical Engineering. https://doi.org/10.1016/j.bbe.2020.08.005

[14] Shibly, K. H., Dey, S. K., Islam, M. T. U., & Rahman, M. M. (2020). COVID faster R–CNN: A novel framework to Diagnose Novel Coronavirus Disease (COVID-19) in X-Ray images. Informatics in Medicine Unlocked, 20. https://doi.org/10.1016/j.imu.2020.100405

[15] Sekeroglu, B., & Ozsahin, I. (2020). Detection of COVID-19 from Chest X-Ray Images Using Convolutional Neural Networks. SLAS Technology. https://doi.org/10.1177/2472630320958376

[16] Apostolopoulos, I. D., Aznaouridis, S. I., & Tzani, M. A. (2020). Extracting Possibly Representative COVID-19 Biomarkers from X-ray Images with Deep Learning Approach and Image Data Related to Pulmonary Diseases. Journal of Medical and Biological Engineering. https://doi.org/10.1007/s40846-020-00529-4

[17] Zhou, J., Jing, B., Wang, Z., Xin, H., & Tong, H. (2021). SODA: Detecting COVID-19 in Chest X-rays with Semi-supervised Open Set Domain Adaptation. IEEE/ACM Transactions on Computational Biology and Bioinformatics. https://doi.org/10.1109/TCBB.2021.3066331

[18] Panwar, H., Gupta, P. K., Siddiqui, M. K., Morales-Menendez, R., & Singh, V. (2020). Application of deep learning for fast detection of COVID-19 in X-Rays using nCOVnet. Chaos, Solitons and Fractals, 138(June), 109944. https://doi.org/10.1016/j.chaos.2020.109944

[19] Zargari Khuzani, A., Heidari, M., & Shariati, S. A. (2021). COVID-Classifier: an automated machine learning model to assist in the diagnosis of COVID-19 infection in chest X-ray images. Scientific Reports, 11(1), 9887. https://doi.org/10.1038/s41598-021-88807-2

[20] Luz, E., Silva, P., Silva, R., Silva, L., Guimarães, J., Miozzo, G., Moreira, G., & Menotti, D. (2021). Towards an effective and efficient deep learning model for COVID-19 patterns detection in X-ray images. Research on Biomedical Engineering. https://doi.org/10.1007/s42600-021-00151-6

[21] Chollet, F. (n.d.). Xception: Deep Learning with Depthwise Separable Convolutions. https://arxiv.org/abs/1610.02357

[22]VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION
https://arxiv.org/abs/1409.1556

[23]Deep Residual Learning for Image Recognition
https://arxiv.org/abs/1512.03385

[24]Densely Connected Convolutional Networks
https://arxiv.org/abs/1608.06993

[25]Learning Transferable Architectures for Scalable Image Recognition
https://arxiv.org/pdf/1707.07012.pdf

[26]MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications
https://arxiv.org/abs/1704.04861

[27]Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (n.d.). *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*.
https://arxiv.org/abs/1602.07261

[28] Holland, John H. "Genetic Algorithms." Scientific American, vol. 267, no. 1, 1992, pp. 66–73. JSTOR, www.jstor.org/stable/24939139. Accessed 1 Sept. 2021.

[29] Stanley, K. O., & Miikkulainen, R. (n.d.). Evolving Neural Networks through Augmenting Topologies. Retrieved from http://mitpress.mit.edu/e-mail

[30] Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (n.d.). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
https://arxiv.org/pdf/1810.04805.pdf

[31] Openai, A. R., Openai, K. N., Openai, T. S., & Openai, I. S. (n.d.). Improving Language Understanding by Generative Pre-Training.
https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[32] Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., … Polosukhin, I. (n.d.). Attention Is All You Need.
https://arxiv.org/pdf/1706.03762.pdf

[33] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., … Houlsby, N. (n.d.). AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE.
https://arxiv.org/pdf/2010.11929.pdf

[34] Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., & Carreira, J. (2021). Perceiver: General Perception with Iterative Attention.
https://arxiv.org/pdf/2103.03206.pdf

[35] M.E.H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M.A. Kadir, Z.B. Mahbub, K.R. Islam, M.S. Khan, A. Iqbal, N. Al-Emadi, M.B.I. Reaz, M. T. Islam, "Can AI help in screening Viral and COVID-19 pneumonia?" IEEE Access, Vol. 8, 2020, pp. 132665 - 132676.
https://ieeexplore.ieee.org/document/9144185

[36] Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Kashem, S.B.A., Islam, M.T., Maadeed, S.A., Zughaier, S.M., Khan, M.S. and Chowdhury, M.E., 2020. Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-ray Images.
https://doi.org/10.1016/j.compbiomed.2021.104319

[37] KerasGA, ahmedfgad, GitHub
https://github.com/ahmedfgad/KerasGA

[38] Perceiver image classification, Keras-io, keras-team, GitHub
https://github.com/keras-team/keras-io/blob/master/examples/vision/perceiver_image_classification.py

[39]Perceiver-PyTorch, lucidrains, GitHub
https://github.com/lucidrains/perceiver-pytorch