Comparative Study Of CNN Architectures

Nirbhay Sharma (B19CSE114), Mayank Raj (B19CSE053), Gautam Kumar (B19EE031)

Index Terms—Resnet, Shufflenet, Efficientnet, Googlenet, Squeezenet, Mobilenet, Python, Pytorch

1 Introduction

We are living the era of deep learning and artificial intelligence and on that path there had been many advancements done so far, so many great methods were proposed to solve a particular problem statement, basically the essence of deep learning are deep CNN architectures that are capable to extract features, predict accurately and that too with a very high accuracy, and there are lot of models out there that perform the same task, so on that line we would like to explore some of the state of the art models and concept behind them, in great detail, so our work would produce a comparative study among various deep CNN architectures on a particular dataset. In this project our major contribution are as follows:

- We study 7 CNN architectures and replicate their implementation from scratch
- We compare the architectures based on various evaluation metrics such as accuracy, precision, recall etc.

The rest of the report is organized as follows. In section 2 we discuss the methods, In section 3 we discuss the Experimentation, the results are presented in section 4 and finally we conclude the report in section 5. The code and results are available at https://github.com/nirbhay-design/CNNAlgosComparison

2 Methods

We study 7 CNN architectures named SqueezeNet [4], ShuffleNet [13], ResNet [2], MobileNet [9, 3], EfficientNet [12], Inception_v3 [11], and GoogleNet [10], and we perform a comparative study among these state of the art architectures, we discuss description of each architecture in detail below.

2.1 ResNet

ResNet also known as residual networks is a deep learning architecture which was proposed in 2015 by Microsoft Research. In order to solve the problem of vanishing and exploding gradients, residual network were introduced. In this network, skip connections are used which helps to backpropagate the error properly by skipping few layers.The approach behind this network is instead of layers learn the underlying mapping, we allow network fit the residual mapping. If any layer hurts the performance of the network, it is skipped by regularisation. The network uses a 34-layer plain network architecture inspired by VGG-19 in which the shortcut connections are added. The shortcut connections then convert the architecture into residual network.

2.2 SqueezeNet

SqueezeNet is a deep learning architecture which is comprised of ”squeeze” and ”expand” layers. The convolution layer has only 1x1 filter which is fed to expand layer which has a mix of 1x1 and 3x3 convolution filters. Fire module comprise of squeeze layer and expand layer together. The input is sent into conv layer followed by 8 fire module layers. Followed by another conv layer and max pooling layer. Dropouts can also be introduced to reduce overfitting. Small size if SqueezeNet makes it ideal for deployment.

2.3 ShufflenNet

The idea of ShuffleNet is to use grouped convolutions, depth-wise convolutions [7], and channel shuffle operation. In grouped convolutions it divides input channels in to groups and apply kernel filters on each of them, in this way it reduces the parameters. In depth-wise convolutions the group size is equal to the number of input channels so that each filter is applied to each channel and output is concatenated. In channel shuffle operation it tries to shuffle the channels and mixed them in between so that features can traverse in between the channels also. The idea behind ShuffleNet is to design a light weight CNN architecture without compromising the accuracy. This is why SuffleNet uses depthwise convolution.

2.4 GoogleNet

The core idea of GoogleNet is to pass the input features to various sizes of filters such as 1×1 filters, 3×3 filters, 5×5 filters, etc and concatenate the output to pass to the next set of operations. Later, in order to decrease the parameters, the 5×5 filters are replaced by 3×3 filters. GoogleNet architecture also uses the concept of auxilliary outputs, which is used in between the architecture to predict the outputs, so it is like the output layer before the output layer, the training for such a model is done as follows. First the output from both the output layers are extracted and then loss function determines everything as follows. p indicates predicted probabilities and t indicates target labels, α is the hyperparameter to provide weight to the auxilliary output, it is generally 0.3

$$L = \alpha \times L_{aux}(p, t) + L_{output}(p, t)$$

at the testing time the auxilliary outputs are discarded and only final output layer is considered

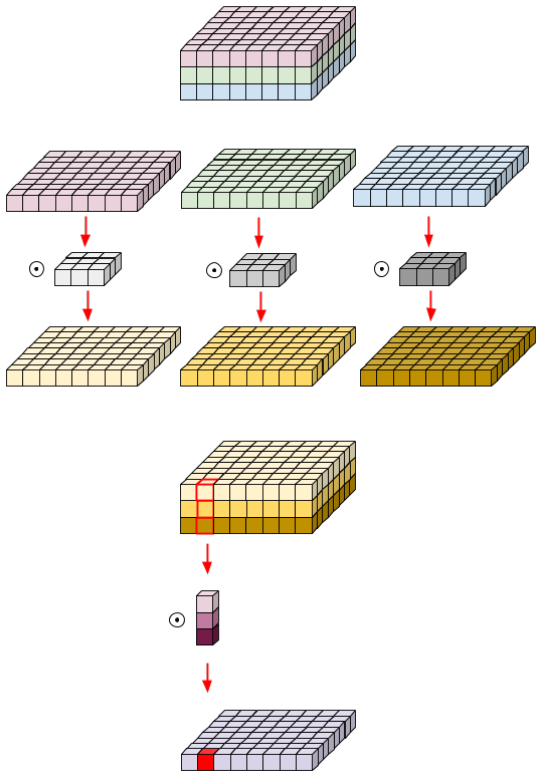

Fig. 1: Representation of depth-wise convolution. The filters are applied on one channel at a time.

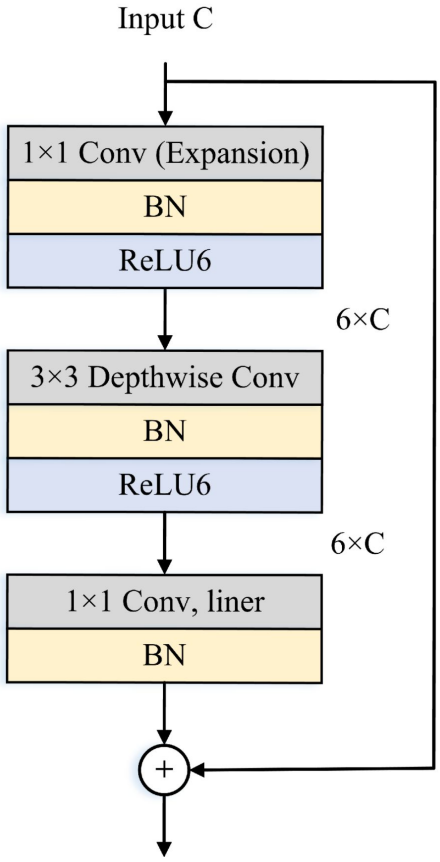

Fig. 2: Representation of MBConv Blocks. Pointwise 1×1 convolution with higher number channels is used for feature extraction, followed by application of depthwise convolution. Finally we have point-wise convolution to reduce the number of channels to assist addition of residual connection.

2.5 MobileNet

MobileNet [8] uses the idea of Inverted Residuals with linear Bottlenecks. As the name suggest, the model was developed to be easily deployed in mobile phones. MobileNet blends the ideas of depth-wise convolution, point-wise convolution and residual connections to form MBConv blocks as shown in Fig 2. These blocks help to reduce the number of multiplication operations, involved in regular convolution operations of residual blocks, by a significant margin. In depth-wise convolution, we apply a single convolutional filter for each input channel. In the regular 2D convolution performed over multiple input channels, the filter is as deep as the input. Depth-wise filter is a a spatial convolution performed independently over each channel of an input as shown in fig 1. Point-wise convolution is a simple scalar multiplication using 1×1 filter, used to scale the number of channels as per the requirement.

2.6 EfficientNet

Through a detailed analysis researchers found that carefully balancing network depth, width and resolution can lead to better performance. Images with higher resolution need deeper networks. Different scaling dimensions are not independent. EfficientNet introduced a new scaling method that uniformly scales all dimensions of depth (d), width (w) and resolution (r). It uses a new compound scaling method, which uses compound coefficient ϕ.

d = α^{ϕ}, w = β^{ϕ}, r = γ^{ϕ}

such that $\alpha.\beta^2.\gamma^2$ = 2. Here ϕ is user specified coefficient that specifies how many more resources are available.

Just like MobileNet, EfficientNet also uses MBConv blocks as shown in fig 2. We have considered EfficientNet-B0 and EfficientNet-B2 for comparison with other models. Both uses = 1.2, β = 1.1 and γ = 1.15, for ϕ = 1. They also use squeeze and excitation optimization as described in section 2.2.

2.7 Inception_v3

The idea of Inception_v3 is to used residual layers with varied types of convolutions operations. It is basically an improvement over Inception_v2, it uses various block such as Inception{A,B,C,D,E} and IncpetionAux block for the auxillary output in between the layers, the architecture of Inception{A,B,C,D,E} is similar to Fig 3 where in each layer there are various branches which has different kernel sizes as 1×5, 5×1, 1×7, 7×1 etc. and the final output from the branches is then concatenated to pass on to the next branch which also has similar architecture, the auxiliary output helps the model to learn the features better and thus increase the performance by a large margin, the loss function equation for Inception_v3 is similar to GoogleNet auxiliary ouptut as

Methods	#Params	Accuracy	Precision	Recall	F1	AUROC
EfficientNet B0	17.55M	0.86	0.79	**0.79**	0.79	0.90
EfficientNet B2	17.55M	**0.88**	**0.84**	0.78	**0.81**	0.89
ResNet 50	23.51M	0.80	0.71	0.54	0.53	0.84
MobileNet V2	2.22M	0.86	0.80	0.78	0.79	**0.91**
SqueezeNet	0.73M	0.82	0.72	0.74	0.73	0.85
ShuffleNet	1.26M	0.85	0.77	0.78	0.78	0.87
GoogleNet	5.6M	0.85	0.81	0.68	0.72	**0.91**
Inception_v3	21.79M	0.79	0.65	0.50	0.45	0.85
Inception_v3_aux	24.35M	0.84	0.80	0.65	0.69	0.86

TABLE 1: Comparative performances through average Accuracy, Precision, Recall, F1 score (F1), AUROC on the test set of Retinal Dataset. $\#Params$ indicates number of trainable parameters in a model. The highest value in each column is highlighted in bold.

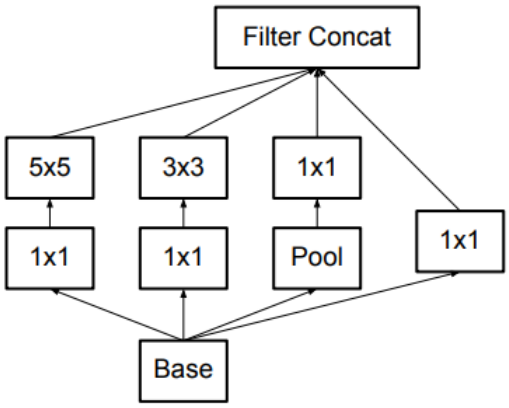

Fig. 3: Sample Representation of a building block architecture of inception_v3.

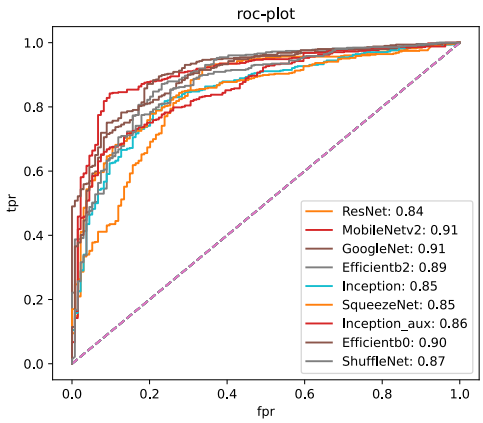

Fig. 4: ROC plot showing the true positive rate (tpr) vs false positive rate (fpr) with AUC values for all the state-of-the-art models.

$$L = \alpha \times L_{aux}(p,t) + L_{output}(p,t)$$

3 Dataset & Experimental setup

We use the training set of Retinal Disease Classification Dataset [5] (abbreviated as Retinal Dataset in report) for training, validation set for validation and test set for testing, the dataset consists of 46 different classes and one No finding class. We confine the problem to binary classification task. We assign label 0 to No finding class and assign label 1 to any retinal disease class i.e. abnormal eye, The dataset consists of 1920 training images, 640 validation images, and 640 images for testing and we are using all the training images for training over 30 epochs, We use a minibatch size of 8 and image size 512 $\times$ 770. We use ADAM [1] optimizer with initial learning rate of 10^{-3} to train the models and cross-entropy as loss function. We use pytorch for implementing models from scratch and python version 3.8.

4 Results

We train all the models and calculate the performances of the model on various evaluation metrics [6], such as Accuracy, Precision, Recall, F1score, AUROC as presented in Table 1. The roc curve for all the models is presented in 4.

4.1 Discussion on Performance

All the models are performing well according to their capabilities the summary of comparison is presented in Table 1. All the highest entries in the column is specified in bold. From the Table, we can infer that EfficientNet B2 is out performing other models in terms of average Accuracy, Precision, F1 score, EfficientNet B0 is out performing others in terms of average Recall score, MobileNetv2, GoogleNet are getting same AUROC values and are outperforming others in terms of AUROC values. Apart from this, we can note that EfficientNet B0,2 have higher number of trainable parameters and GoogleNet with less number of parameters (5.6M), outperforms EfficientNet models in terms of AUROC values with a small margin. Some models such as Inception_v3 has a very high number of parameters but it not performing that good and on some metrics such as Recall, F1 score. It's performance on vision dataset is low, which indicates that it does not generalize well and is biased to one specific class and hence not a suitable model for this applicaiton. One thing to note here that the inception with auxiliary output performs much better than the same inception model which has auxiliary output disable, this clearly shows that the auxiliary output helps the model to generalize and learnt well. In the nutshell, all the models have performed good with limited dataset of only 1920 train images and were able to generalize well.

5 Conclusions & Future work

So far in the report we have seen the performance of various state-of-the-art architectures such as EfficientNet, MobileNet, ResNet, InceptionNet etc. and their ability to learn on the Retinal Dataset in a better way. The above mentioned state-of-the-art models marks only the beginning of the vast field of machine learning. There are lots and lots of models coming every day with new advancement and achieving / beating current state-of-the-art architectures. The above comparison study on few of the architectures shows the advancement in machine learning field and how the models evolve from AlexNet to ResNet to EfficientNet and so on. Based on the day to day advancement in the field of machine learning the future works of this project is unbounded, various models can be compared with one another on the basis of various tasks such as object detection, image segmentation, image generation etc. and blending the ideas of models is also possible to come up with an altogether new model which is an experienced combination of the state-of-the-art methods and which can also beat the current state-of-the-art models as well. So the extensions to this project are endless.

6 Acknowledgement

The members of the team would like to thank the instructor, Dr. Mayank Vatsa, for giving us the opportunity to work on such an interesting project.

References

[1] Sebastian Bock, Josef Goppold, and Martin Georg Weiß. "An improvement of the convergence proof of the ADAM-Optimizer". In: *CoRR* (2018).

[2] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015).

[3] Andrew Howard et al. "Searching for MobileNetV3". In: *CoRR* (2019). arXiv: 1905.02244. URL: http://arxiv.org/abs/1905.02244.

[4] Forrest N. Iandola et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡ 1MB model size". In: *CoRR* abs/1602.07360 (2016).

[5] Samiksha Pachade et al. "Retinal Fundus Multi-Disease Image Dataset (RFMiD): A Dataset for Multi-Disease Detection Research". In: *Data* 6.2 (2021). ISSN: 2306-5729. DOI: 10.3390/data6020014. URL: https://www.mdpi.com/2306-5729/6/2/14.

[6] David M. W. Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". In: *CoRR* abs/2010.16061 (2020).

[7] Zheng Qin et al. "Diagonalwise Refactorization: An Efficient Training Method for Depthwise Convolutions". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–8.

[8] Mark Sandler et al. "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation". In: *CoRR* abs/1801.04381 (2018). arXiv: 1801.04381. URL: http://arxiv.org/abs/1801.04381.

[9] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[10] Christian Szegedy et al. "Going Deeper with Convolutions". In: *CoRR* abs/1409.4842 (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[11] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.

[12] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[13] Xiangyu Zhang et al. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.