EVALUATING THE PERFORMANCE OF POPULAR NEURAL NETWORK ARCHITECTURES ON THE CIFAR-10 DATASET

By

Gagandip Chane, BASc, University of Waterloo, 2013-2018

Mathusan Thanabalasingam, BASc, University of Waterloo, 2014-2019

A Term Project

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Masters of Science (MSc)

in the program of

Data Science & Analytics

Toronto, Ontario, Canada, 2020

Author’s Declaration

We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed.

Gagandip Chane

Mathusan Thanabalasingam

Abstract

This report explores several iconic architectures and tests their performance on the popular CIFAR-10 image classification dataset. The paper covers literature review regarding the difference between some of these models. The paper also covers literature review regarding the use of transfer learning and ImageNet weights on a dataset completely unrelated to the ImageNet dataset. This paper also experiments with transfer learning, and tests to see if transfer learning using ImageNet weights would increase the performance of some of these models. Analysis showed that transfer learning did not benefit all models; however, the model with the best combination of validation accuracy and training time, was the InceptionV3 model that utilized transfer learning and ImageNet weights.

Table of Contents

List of Tables

List of Figures

Introduction

A major application for machine learning is image classification. The ability to classify the contents of an image using a model can be used in all industries. Image classification problems can vary from identifying an Alzheimer's disease patient through their MRI brain scan, to understanding weather patterns through satellite images.

A commonly used Neural Network for image classification is a Convolutional Neural Network (CNN). The hidden layer of a CNN includes multiple convolutional layers, which basically take similarly colored pieces or edges of the pixels, and make larger pieces out of them, like how you would assemble a puzzle. This is very useful for an image recognition problem.

The CIFAR-10 dataset is a commonly used set of data to evaluate the effectiveness of image classification neural networks. The purpose of this report is to compare the performance of iconic architectures on the CIFAR-10 dataset. We also try to compare the effectiveness of transfer learning with some of these iconic architectures that were trained using the ImageNet database, and see if these weights can translate to the CIFAR-10 dataset.

We also look at the effect of optimizing the actual dataset and seeing if transformations to the data could lead to greater accuracy in our models. We will compare our findings with state-of-the-art models that were designed for the CIFAR-10 dataset, as well.

Data Sources

The CIFAR-10 dataset is an established image classification dataset from Google developers Geoffrey Hinton, Alex Krizhevsky and Vinod Nair [1]. For this project, however, we were able to obtain the dataset from Keras, a python library specifically for machine learning.

There are two versions of the CIFAR dataset: CIFAR-10 and CIFAR-100. Due to lack of computational resources, we decided to attempt to optimize the CIFAR-10 dataset rather than the latter. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images, and there are an equal number of images for each class.

The 10 classes, or types of images, are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. It should be noted that all classes are distinctly exclusive; a truck is an automobile, but all truck images have been labeled as trucks, and no Automobile-defined image is a truck. Sample images can be viewed in Figure 1.

Figure 1: Sample Images from the CIFAR-10 Image dataset [1].

Literature Review/Related Work

The CIFAR-10 dataset is much simpler than that of a much larger and more complex dataset called ImageNet [2]. The ImageNet dataset consists of 14 million color images that fall into one of one thousand classes. These images are also of much higher resolution; where the CIFAR-10 images have dimensions of 32 x 32, the ImageNet images average a 469 x 387 resolution. Several iconic architectures were formed to tackle the ImageNet classification problem, and it was noted that with the diversity of the ImageNet dataset, it would be possible to transfer the weights obtained by these architectures, for other image classification problems. An example of this can be observed in [3].

This research paper from Ryerson University explored the performance of the Inception v4 model and VGG-16 model with classifying MRI images belonging to patients that either have or do not have Alzheimer's Disease. The two models were pre-trained with the weights these models learned from the ImageNet dataset, with the only exception being for the fully connected layers at the end of each respective convolutional neural network. The output layer was adjusted to only include the necessary classes; in the case of the paper, the patient either had or did not have Alzheimer's disease, so there was one output node, activated by a sigmoid function.

The pretrained models were only having the weights adjusted for the new output layers. The two pretrained models were evaluated against a VGG-16 model with no prior training. The results of the experiment showed that the two pretrained models, although having layers in the early stages of the network trained on a completely different dataset, performed significantly better than the VGG-16 model where all layers needed to be trained. The Inception model yielded an accuracy score of 96.25%, while the pretrained and not pretrained VGG-16 models yielded accuracy scores of 92.3% and 74.12%, respectively.

This paper also discusses the use of image entropy calculations, which were used to significantly reduce the size of the dataset. The OASIS dataset, an official database of MRI scans, consists of over 100 thousand images; the training set used in the experiment was approximately 5,000. This shows that transfer learning, as well as selecting the most informative images, allowed for an extremely high accuracy while being able to dramatically reduce training size. For the purpose of keeping our experiment as controlled as possible, we will use the same data to train all the models.

Other related work explained four famous CNN architectures and how they differ from each other [4]. The first architecture discussed is AlexNet, which was a deep convolutional network built to improve the result on the ImageNet challenge, achieving 84.7% accuracy. The network consisted of 5 convolutional layers and 3 fully connected layers. On the ImageNet dataset, data augmentation including image mirroring and cropping was performed to increase the variation in data and reduce overfitting. Overlapping maxpool layers were added after certain convolutional layers, resulting in slight improvement in the error rate. ReLU was used as the activation function in the network instead of the commonly used sigmoid and tanh functions which suffered from the vanishing gradient problem. Lastly, dropout layers were added to further prevent overfitting, however this caused the number of iterations till convergence to increase.

The next architecture discussed is VGGNet. The key element in this architecture is that the kernel size is fixed at 3x3 as compared to differing kernel sizes in AlexNet. As a result, the number of parameters to learn in convolution layers is reduced. By using multiple smaller 3x3 kernels, the receptive field covered by larger kernel sizes is replicated. This enables the network to learn faster and is less prone to overfitting.

ResNet, meaning residual networks, was the third architecture that was discussed. Different variants of ResNet include ResNet18, ResNet50, and ResNet101. These networks started to get deeper since the vanishing gradient problem was dealt with using a different method

here. Like VGGNet, 3x3 kernel size was used throughout and pooling was only performed twice, once after the first layer and then after the last convolutional layer. The unique element here is that identity connections are used between every two convolutional layers. This means the output from the layer before the convolutional layer is combined with the output of the convolutional layer, this solves the vanishing gradient problem and is the essence of the residual block which is repeated throughout the network.

The last architecture discussed is Inception. The idea behind this network is to use kernels of different sizes that are used within the same layer in parallel, hence the network goes wider instead of deeper. This captures global features using large kernel sizes and area specific features using small kernel sizes, within the same layer. Features from different kernels are concatenated before being fed to the next layer. During training and backpropagation, the network decides which features are most important based on the images at hand and assigns weights accordingly.

Table 1 shows the improvement in classification accuracy over the years as these famous architectures were developed.

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

Table 1: Classification Accuracy of Iconic Architectures. [4]

Technologies Used

The entire experiment will be executed in Python3. The most important technology to note here is the use of the tensorflow.keras package within Python. This package had the CIFAR-10 dataset preloaded, as well as all the tools required to create a Neural Network. This package also had some of the iconic architectures already generated as saved as their own objects, and these were leveraged for use in this report as well. There was the use of the skimage library to assist in changing the resolution size of the dataset, which will be further explained in the preprocessing section of this report.

Issues Faced

There were some issues that were worked around as we completed the entirety of this project. The first of these problems was being unable to replicate the exact training and validation errors being obtained by the Keras version of the VGG16 model, and our manually generated version of the model.

Another problem we faced was that the InceptionV4 model was not a model already available in the keras.applications library; the most recent version of Keras found in this library was InceptionV3. InceptionV4 was quite complex to manually generate, and there were several other models for us to compare with anyways. For this reason, we decided to move forward with the InceptionV3 model.

Another issue that we ran into was the input shape for the InceptionV3 model. Although the CIFAR-10 dataset images are 32 x 32 resolution for height and width, the Inception model required them to be a minimum of 75 x 75. Having to resize all the images was a bit of a time-consuming process and crashed the python environment when we attempted to resize to a higher resolution, to test if image size would affect model performance for the other models. We decided to only resize the images for use in the InceptionV3 model.

Data Manipulation and Pre-Processing

Prior to creating and evaluating the models, the dataset had to be manipulated for use in the models. Each image in the CIFAR-10 dataset has three channels, one for each of the colors in the RGB color model. These numbers are integers and need to be converted to float for use in the models. Another key type change that needed to occur was for the target variable: the class of each image. In the original dataset, the target variable was an integer between 0 and 9, to represent one of the ten possible classes the image could belong to. As we know that the output layer of all the neural networks are activated using a SoftMax function, the output would need to be a list of 10 integer values, where 9 of them are 0s and the index representing the class is marked as 1. We convert the class variable from integer to categorical, using the np_utils function in Keras.

As mentioned in the Issues section, we needed to resize the images in order to accommodate the InceptionV3 model. The CIFAR-10 images were resized from 32 x 32, to 75 x

75. We chose 75 x 75 because that is the minimum resolution that could be accepted by the Inception model, and attempting larger resolutions was computationally expensive.

We also chose to include transformations outside of resizing into our dataset for each image. The ImageDataGenerator in Keras can generate more data based on the CIFAR-10 dataset and will include transformations specified by the user. We decided to include randomly rotated images, horizontal flips, and random shifts of the images, both vertically and horizontally. The hope was that this would allow the models to learn two images that are identified under the same class but could be slightly varied.

Analysis and Results

After processing the CIFAR-10 dataset, multiple deep learning architectures were leveraged to solve the task of classifying the images. All the architectures were based on convolutional neural networks. There have been many advancements made in image classification using convolutional neural networks and this yielded some popular architectures including LeNet [5], AlexNet [6], VGG-16[7], ResNet [8], and Inception [9]. These architectures were trained on the popular ImageNet dataset and provided good results during testing. In the analysis shown below, these architectures or a variation of these architectures were leveraged to train deep learning models on the CIFAR-10 training set.

The first architecture designed was LeNet-5 which was released in 1998 and pioneered the work on image classification. This model consists of 7 layers and was trained on the popular MNIST dataset. Figure 2 shows the network architecture for LeNet-5.

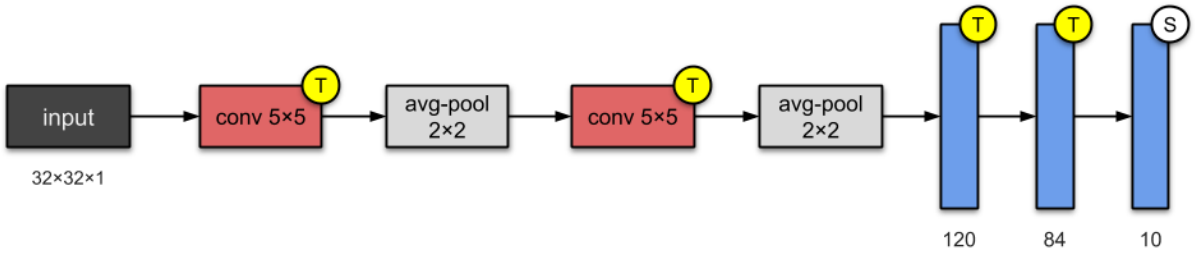

Figure 2: LeNet-5 Architecture Diagram. [10]

The architecture consists of two convolution layers with 6 and 16 filters, respectively with the filter size being 5 by 5. Average pooling layers of size 2 by 2 is added after each convolution layer. As in with all CNNs the output of the convolution and pooling layers is flattened before being

fed into the fully connected layers. There are two dense layers with 120 and 84 hidden nodes, followed by the output layer that gives the prediction as a probability across the 10 classes. All the layers use tanh activation aside from the output layer which is SoftMax. Tanh is not used much in recent architectures and it is also computationally more expensive. Applied on the CIFAR-10 dataset, a validation accuracy of 57% and a training accuracy of 55% was achieved after 50 epochs. Both the training and validation accuracies were still trending upwards after 50 epochs and can be viewed in Figure 3. As will be further explained in the upcoming architectures, LeNet was not able to learn the different patterns and shapes very well, since the training accuracy didn't even reach 60% shows that this model is too general when it comes to classifying RGB images into 10 classes. It's unlikely that an increase in the number of epochs would result in a significantly greater validation accuracy score, as the slope of the curve is already appearing to be flattening.

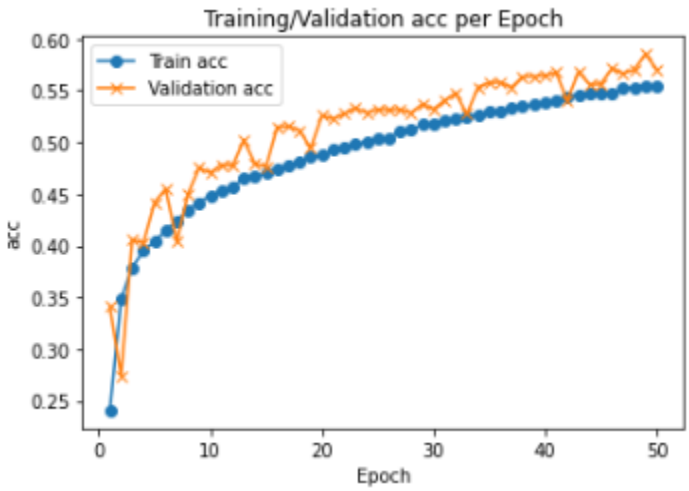

Figure 3: Training and Validation Accuracies for LeNet on the CIFAR-10 Dataset.

LeNet fared well on the MNIST dataset, but one of its drawbacks was that it was an excellent model only to classify images with a grayscale colour scheme. The ILSVRC was a competition focused on accurately classifying the ImageNet dataset. The first winning entry for that competition in 2012, was named the AlexNet model. Figure 4 displays the AlexNet architecture.

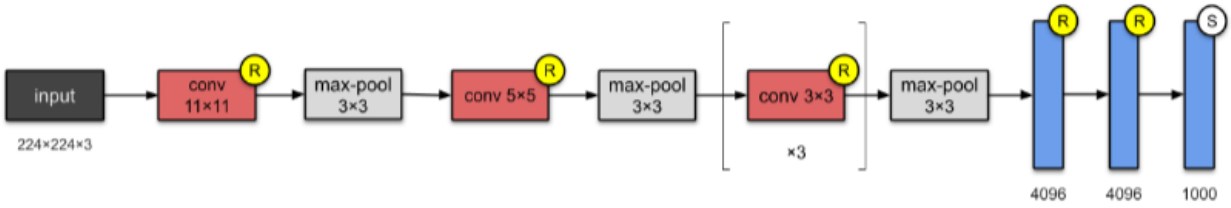

Figure 4: AlexNet Architecture Diagram. [11]

The AlexNet model followed the same sort of format in terms of block patterns as LeNet but was a deeper network and used max pooling rather than average pooling. The other distinction is that this model used the 'ReLU' activation function and applied a 0.5 dropout after each fully connected layer at the end. ReLU activation is computationally more efficient which makes it useful as networks get deeper. The training process for AlexNet on CIFAR-10 is shown in Figure 5. It is evident that the variance between the training and validation accuracy is not as high as LeNet. This is due to the dropout layers that are added, however only after the fully connected layers which has reduced overfitting.

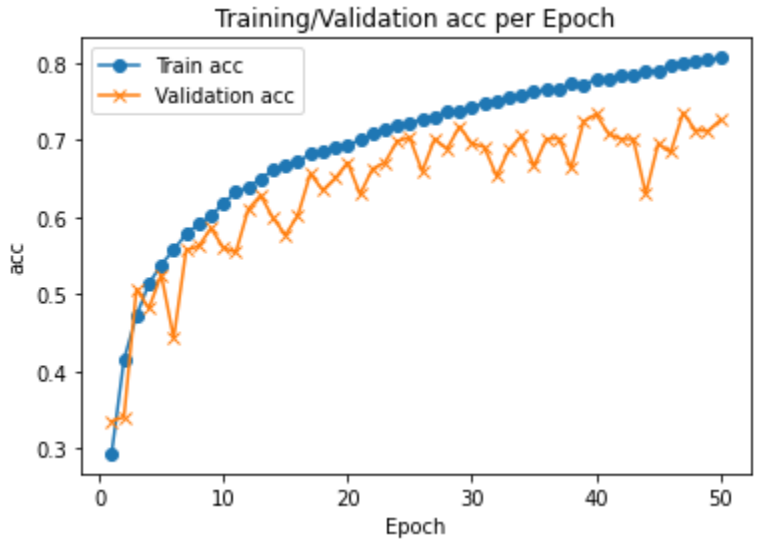

Figure 5: Training and Validation Accuracies for AlexNet on the CIFAR-10 Dataset.

The validation accuracy has increased to approximately 73% after 50 epochs with the training accuracy reaching 81%. This shows that using a deeper architecture and increasing the number of parameters to learn allows more shapes and patterns to be learned.

The next model attempted was the VGG-16 model. This model had improved upon AlexNet's accuracy on the ImageNet dataset. Figure 6 shows the architecture for VGG-16.

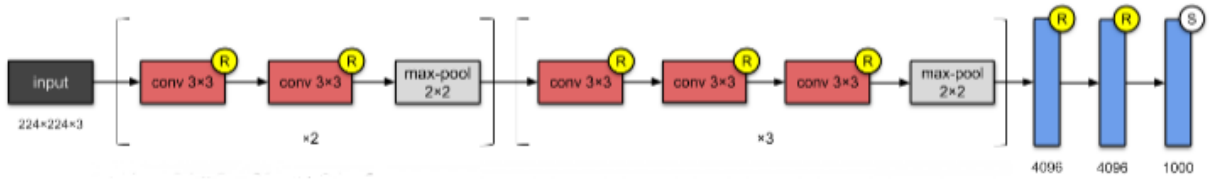

Figure 6: VGG-16 Architecture Diagram. [12]

A key feature of this model is that within each block, as identified by the square brackets, the number of filters increase by a factor of two, while the feature map size decreases by a factor of two. This allows for the model to differentiate between more basic lines and features at the beginning, and form shapes as you go deeper into the network, like a jigsaw puzzle would work. This model, except for the output layer, strictly uses the 'ReLU' activation function at each layer. To compare the effect of transfer learning, the VGG-16 model was created two times; once with pretrained weights from ImageNet and frozen CNN layers, and one without the pretrained weights. Both models were executed using the regular 32 x 32 images. The best validation accuracy achieved with VGG-16 was 85% after 50 epochs through the model that was trained without transfer learning. Figure 7 shows the validation accuracy for the training process for VGG-16 without transfer learning.

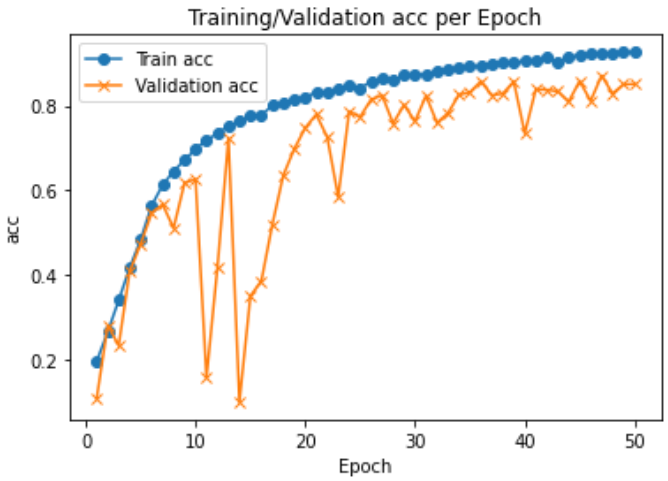

Figure 7: Training and Validation Accuracies for VGG-16 on CIFAR-10 without Transfer Learning.

When evaluating the VGG-16 model that utilized transfer learning, we decided to freeze the CNN layers, and only focus on training the fully connected layers. The expectation was that the CNN layers have already been trained to identify certain features and shapes within each image. With those layers frozen, the number of trainable parameters was reduced by 75%, so we were expecting training time to be drastically less than that of the "trained from scratch" VGG-16

model. To our surprise, not only was it only marginally faster (25 minutes to 28 minutes), but as apparent in Figure 8, the training and validation accuracies are much lower in the transfer learning model, grading at 66% and 63%, respectively. It appears the validation accuracy plateaued very early on, around the 20th epoch. Hence, VGG-16 without transfer learning was chosen as the best one.

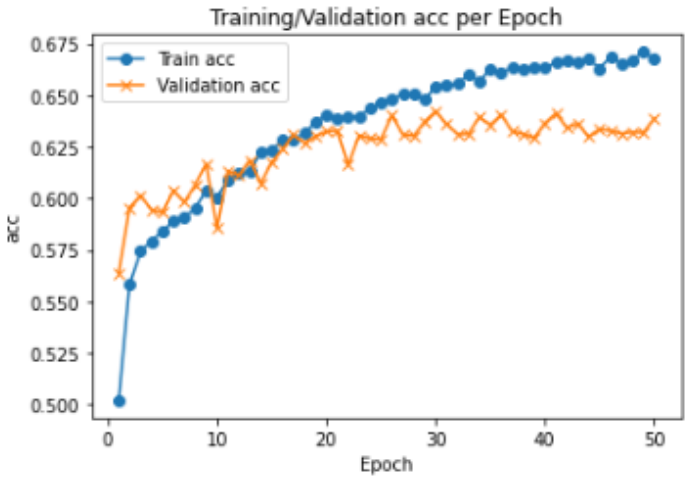

Figure 8: Training and Validation Accuracies for VGG-16 on CIFAR-10 with Transfer Learning.

In 2015, the ILSVRC was won by a new model called ResNet, that introduced the idea of residual layers. The underlying idea of a residual layer is to have weights associated with layers deeper in the network.

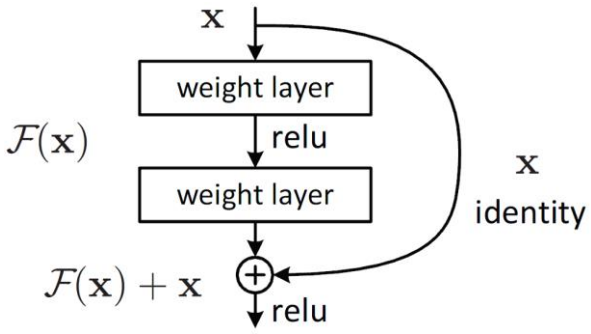

Figure 9: Visual Depiction of Residual Block [13].

The goal of using residual weights was to solve the vanishing gradient problem, which the model proved to have done quite well with the ImageNet dataset. The ResNet model had 152 layers when it was tested on the ImageNet dataset. For the CIFAR-10, the pre-built ResNet50 architecture was leveraged. As was done with VGG-16, we compared a ResNet model trained from scratch, with one that used ImageNet weights and had frozen CNN layers for transfer

learning. The training progression of the ResNet model trained from scratch can be viewed in Figure 10 below. The graph shows random dips in performance, both in earlier and later epochs, with the validation accuracy reaching 79% by the last epoch.

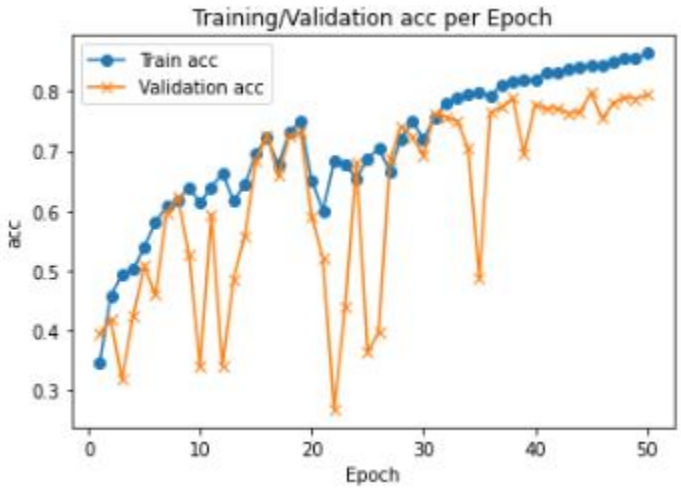

Figure 10: Training and Validation Accuracies for ResNet on CIFAR-10 without Transfer Learning.

Implementing transfer learning on the ResNet model yielded a much lower accuracy, as can be seen in Figure 11. Using pre-trained weights resulted in the performing quite well in the start of training but failed to improve at the rate that the other ResNet model did, only reaching 64%. There was clearly a heavy case of overfitting, as the training accuracy continued to increase while validation accuracy stagnated after the 15th epoch.

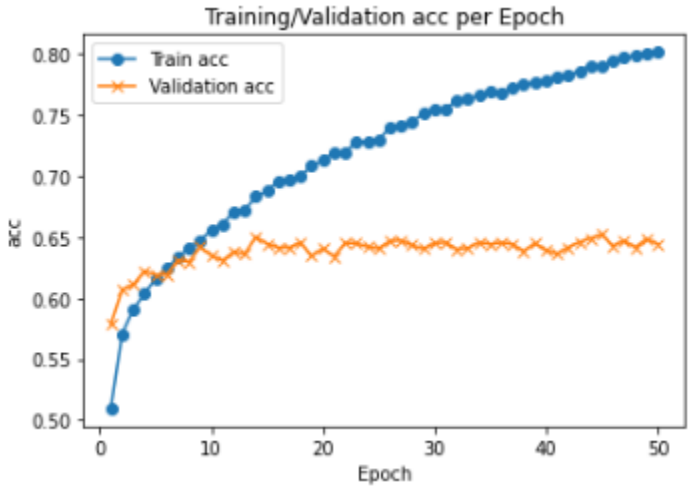

Figure 11: Training and Validation Accuracies for ResNet on CIFAR-10 with Transfer Learning.

Although the transfer learning model was a lot less accurate, it should be mentioned that it was still more time-efficient in comparison to the one without transfer learning. The ResNet

transfer learning model took roughly 33 minutes to train over 50 epochs, whereas the former model took a little under one hour to train. However, the trade-off in time is not enough to justify the significant validation accuracy drop, so the ResNet model without transfer learning is chosen as the best one.

The final architecture that will be demonstrated is Inception v3. Version 1 of this model won the 2014 ILSVRC competition and has now been released with minor modifications four times. We chose to use Inception v3 because creating any of the models from scratch would be a time-consuming process, and Keras offers Inception v3 with and without the weights collected from training on the ImageNet dataset. The building block of the Inception architecture can be seen in Figure 12.

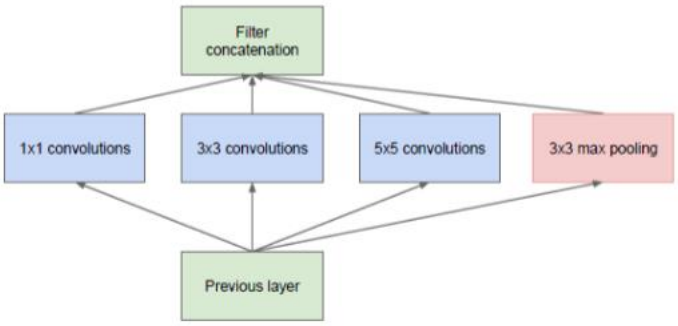

Figure 12: Visual Depiction of an Inception Block [14].

The idea is to use multiple kernel sizes for each convolutional layer instead of just one size. This allows bigger filter sizes to capture global features of images while the smaller kernels capture the local features. In applying this, the network gets wider instead of deeper. We compared the Inception v3 model with pretrained weights from ImageNet, once with no layers frozen, and another with frozen CNN layers for transfer learning, and evaluated their performance on the CIFAR-10 dataset. As previously mentioned, the images for this architecture were enlarged to 75x75, meaning more parameters must be learned compared to the original image size of 32x32. Due to the high training time, this architecture was only trained for 20 epochs. Despite the decreased number of epochs, the Inception model without transfer learning achieved a validation accuracy of 80%, greater than all models that have been compared so far, other than the VGG-16 model. The training progression of this Inception model can be seen in Figure 13.

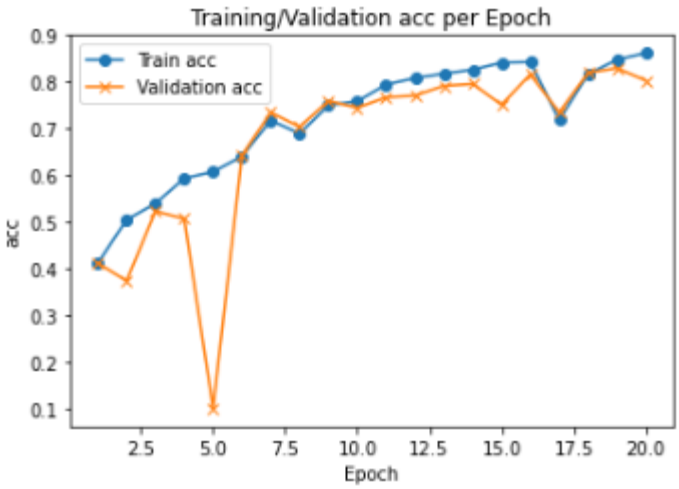

Figure 13: Training and Validation Accuracies for Inception v3 on CIFAR-10 without Transfer Learning.

While the transfer learning model for VGG-16 and ResNet resulted in lower accuracies than their “trained from scratch” counterparts, Inception yielded an improvement. With the Inception transfer learning model, validation accuracy reached 88%. This was the highest accuracy score recorded by any of the models. The training progress can be viewed below in Figure 14.

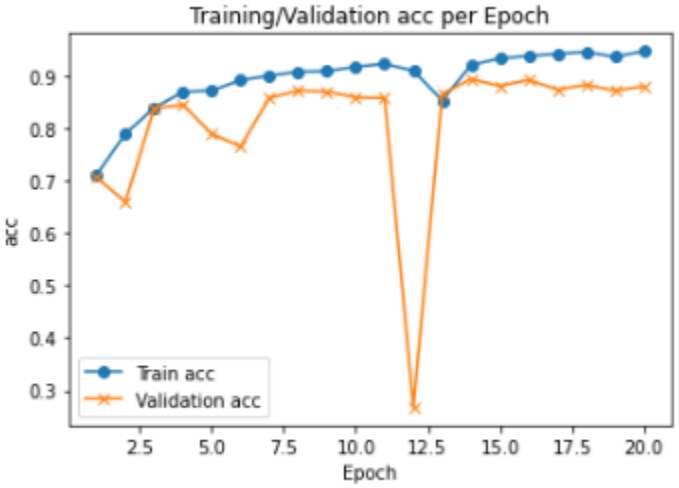

Figure 14: Training and Validation Accuracies for Inception v3 on CIFAR-10 with Transfer Learning.

The starting validation accuracy was near 70%, so over the training processes the increase in validation accuracy was not as significant, but 88% was best overall. Additionally, both Inception models experienced a random dip in accuracy, but recovered in subsequent epochs. This experiment demonstrates that a wider network may be more suitable on CIFAR-10 dataset compared to a deeper network such as ResNet.

Another reason that the Inception transfer learning model should be favored over the from-scratch model is the significant decrease in training time. While achieving a higher accuracy score, transfer learning for the Inception model was completed in 60% of the time (23 minutes vs. 38 minutes). This can be explained by a significant amount of weights staying constant, so there were less parameters that needed to be adjusted over the 20 epochs. The table below shows the optimal training time for each architecture. However, Inception, when utilizing transfer learning, achieved the highest accuracy and required fewer epochs to train than every other model, despite individual epochs taking longer.

Architecture	Epochs	Training Time (minutes)	Validation Accuracy
LeNet (Manual)	50	24	57.1%
AlexNet (Manual)	50	25	72.6%
VGG-16 Pre-Built No Transfer Learning	50	19	85.1%
VGG-16 Pre-Built Transfer Learning	50	18	63.7%
ResNet Pre-Built No Transfer Learning	50	50	79.4%
ResNet Pre-Built Transfer Learning	50	33	64.5%
Inception v3 No Transfer Learning	20	57	80.3%
Inception v3 Transfer Learning	20	37	88.0%

Table 2: Experimental Results on the CIFAR-10 Dataset

Table 2 summarizes the performance of different models. Overall, the Inception network with its wide architecture performed the best while leveraging transfer learning. ResNet did not perform as expected given its deep architecture and the training time was also comparatively high. VGG-16 without transfer learning seems to be the most efficient as the training time was under 20 minutes, achieving a solid 80% in validation accuracy.

It should be noted that experiments were attempted with two different training sets; one that required the use of the ImageDataGenerator library allowing augmentation, and the second with the original dataset with no transformations. For the sake of focusing on more important points of analysis, we chose to omit the observations made without using the ImageDataGenerator library, as every result in those models were worse than what we observed when applying augmentations.

Comparisons to State-of-The-Art Models

The first comparison is to the model FastAutoAugment [15], which incorporates a fast auto-augment procedure to find optimal augmentation policies followed by training on various networks such as Wide-ResNet, Shake-Shake, and PyramidNet+ShakeDrop. AutoAugment is an algorithm that finds the most effective Augmentation parameters for a dataset but is computationally expensive. FastAutoAugment introduced a more efficient search algorithm in order to achieve significantly lower training time. The best resulting accuracy on CIFAR-10 was 98.3%, which is about 9% higher than the Inception model shown in this report. By focusing on improving the training set, significant increase to performance can be achieved.

Another comparison has been done with a variation of ResNet and is titled Wide Residual Networks [16]. Instead of going deeper, this paper chooses to opt depth for width of network, creating a 16-layer wide residual network. This change yielded a superior performance; 96.11% accuracy was achieved by using this technique on CIFAR-10, which is about 7% higher than our Inception model. Both comparisons were done with implementations that did not use additional training data to account for fairness.

Conclusions

The above analysis shows that of all the architectures used, the InceptionV3 model proved to be the most effective architecture for classifying the CIFAR-10 dataset in terms of accuracy, achieving 88%. Transfer learning using the ImageNet weights proved to be quite beneficial to the Inception model as well, as it maintained its accuracy while significantly reducing the training time. However, our evidence shows that transfer learning was not successful in increasing the accuracy

for all the networks it was implemented for and could be something to investigate. Future expansions on this analysis could include using a machine with higher computational power to test the improved Inception model, titled Inception V4, which requires images to have a resolution of at least 299 x 299.

Lessons Learned

Based on the analysis performed, as well as research into some of these state-of-the-art models, we noticed that one of the best ways to improve the accuracy of a model is to modify the actual data, rather than the models. Initially ImageDataGenerator was not considered, but upon learning about its functionalities, it looked promising. Although it made every model slightly slower, it still managed to increase the accuracy of every model. Conceptually it makes sense, as the model is now being introduced to transformations of images which capture the different variations of images present within classes. This allowed for the model to generalize and better fit for a wider range of images. The first state-of-the-art model we described focused specifically on the optimal augmentation measures that can be applied to the training data, and the resulting models ended up being significantly more accurate than most state-of-the-art models created for this exact problem.

It was interesting to see that transfer learning did not work as well as we had expected it to for the VGG-16 model. Based on the paper focused on applying the same ImageNet weights to a set of MRI brain scans, we expected the results of the transfer learning models to be significantly higher than that of the models without transfer learning. It was almost identical for Inception, albeit significantly faster, but VGG-16 was not faster nor more accurate.

Contributions

Gagan created the LeNet, AlexNet and ResNet models that were evaluated in this report and conducted all the analysis for these three models. Gagan also did the analysis regarding comparison to the state-of-the-art models, as well as the Conclusions section. Gagan also did literature review for the article explaining the difference between AlexNet, Resnet, VGG and Inception.

Mathusan created the VGG-16 and Inception models and conducted all the analysis for these two models. Mathusan wrote the introduction, technologies used, preprocessing, and lessons learned sections. Mathusan also did literature review for the article regarding the application of transfer learning.

Both Mathusan and Gagan formatted and cited their own sections.

References

[1] Krizhevsky, Alex. Learning Multiple Layers of Features from Tiny Images. https://www.cs.toronto.edu/~kriz/cifar.html.

[2] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.

[3] M. Hon and N. M. Khan, "Towards Alzheimer's disease classification through transfer learning," *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Kansas City, MO, 2017, pp. 1166-1169.

[4] Anwar, Aqeel. “Difference between AlexNet, VGGNet, ResNet and Inception.” Www.towardsdatascience.com, 7 July 2019, towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96.

[5] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[6] Krizhevsky, Alex E., et al. “ImageNet Classification with Deep Convolutional Neural Networks.” Neural Information Processing Systems, 2012.

[7] Simonyan, Karen, and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” ICLR 2015, 2015.

[8] He, Kaiming, et al. “Deep Residual Learning for Image Recognition.” 2015.

[9] Szegedy, Christian, et al. “Rethinking the Inception Architecture for Computer Vision.” 2015.

[10-12] Karim, Raimi. “Illustrated: 10 CNN Architectures.” *Www.towardsdatascience.com*, 29 July 2019, towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d.

[13] https://miro.medium.com/max/1000/1*6HDuqhUzP92iXhHoS0Wl3w.png

[14] Tsang, Sik-Ho. “Review: GoogLeNet (Inception v1)- Winner of ILSVRC 2014 (Image Classification).” *Medium*, Coinmonks, 1 June 2019, medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7.

[15] Lim, Sungbin, et al. “Fast AutoAugment.” *Neural Information Processing Systems*, 2019.

[16] Zagoruyko, Sergey, and Nikos Komodakis. “Wide Residual Networks.” 2016.