Deep Learning Project

Group Members:
Farooq Mahmud

October 11, 2024

1 Introduction

Deep learning models, particularly Convolutional Neural Networks, have made great strides in solving complex computer vision tasks such as image classification. Among the key advancements in this area is the Inception architecture, introduced by Szegedy et al. (2015) in the paper "Going Deeper with Convolutions." This architecture was designed to balance computational efficiency with improved classification accuracy. This architecture achieved winning results in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2014. The primary innovation of the Inception architecture is its use of multi-scale feature extraction and dimensionality reduction, which allows it to increase network depth and width without excessive computational overhead [13].

This paper aims to replicate the results presented by Szegedy et al., critically assess the impact of the Inception architecture, and test its performance on an alternative dataset. We attempt to reproduce the GoogLeNet model by writing our own Python implementation using the Tensorflow library. Our implementation is evaluated on both a subset of the ImageNet dataset and a secondary dataset, CIFAR-10, to assess its generalization capability. Through this replication and exploration, we seek to validate the core contributions of the Inception model, gain a deeper understanding of convolutional neural networks, and an appreciation of the associated challenges.

2 Motivation

The primary motivation for attempting to reproduce the experiment from the Szegedy paper was to apply TensorFlow to a practical problem, such as image classification, using a realistic dataset. The detailed documentation of the network in the Szegedy paper greatly facilitated its TensorFlow implementation. Recognizing that the operational environment would be constrained (i.e., running the reproduction on a desktop PC), we also saw this as an opportunity to evaluate the network's performance on a smaller version of the ImageNet dataset. This project provided a hands-on experience with the challenges of

training a neural network. Additionally, it offered valuable experience in writing TensorFlow code for a complex neural network with branching and joining structures, a significant leap from our prior work with simpler sequential networks. For comparison, we also experimented with a sequential network to highlight the differences in complexity and performance.

3 Inspiration

Two code bases served as the inspiration for our Tensorflow implementation of Inception. The image pre-processing, particularly regularization techniques were guided by [9]. The Inception layer implementation was inspired by [2]. Lastly, the graphs in Figures 3 and 5 were inspired by [3].

4 Background

GoogLeNet introduced a novel approach to convolutional neural networks by stacking multiple Inception modules. Each Inception module consists of parallel branches that apply different-sized convolutional filters (1x1, 3x3, 5x5) and max-pooling operations in parallel. These branches are concatenated to capture multi-scale features.

The Inception module aims to mitigate computational costs by including 1x1 convolutions for dimensionality reduction before performing the larger convolution operations. This was one of the key innovations that made GoogLeNet both efficient and powerful, despite being deeper than previous CNN architectures such as AlexNet and VGGNet [8, 12].

4.1 Inception Architecture

The Inception architecture was introduced to address the primary challenge in convolutional neural networks- improving model accuracy while maintaining computational efficiency. Szegedy et al. proposed a novel structure called the Inception module, which incorporates parallel convolutional filters of different sizes (1x1, 3x3, and 5x5) along with max-pooling. The key advantage of this approach is that it allows the network to capture features at multiple scales simultaneously, enhancing the model's ability to learn complex spatial hierarchies within images.

Another major innovation was the use of 1x1 convolutions for dimensionality reduction. By applying 1x1 filters before 3x3 and 5x5 convolutions, the Inception module significantly reduces the number of parameters and computational cost without sacrificing accuracy. This strategy enables the network to be deeper and wider without a substantial increase in computational complexity.

The Inception architecture was implemented in a 22-layer deep network, GoogLeNet, which was used to win the ILSVRC 2014 competition. As Fig-

ure 1 shows, GoogLeNet has the lowest top-5 error rate compared to previous winners [13].

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Figure 1: GoogLeNet performance at ILSVRC 2014

5 Methodology

5.1 Image Selection

The ImageNet dataset contains over 1 million images with a size of over 1TB. Processing this amount of data would not be practical given the available computing power. In order to make processing possible in a reasonable time frame, a subset of the ImageNet images was downloaded. The subsets, or *synsets* map to the CIFAR-10 image classes as shown in Table 1.

Synset ID	Equivalent CIFAR-10 Class
n02691156	airplane
n02958343	automobile
n01503061	bird
n02121808	cat
n02419796	deer
n02084071	dog
n01641577	frog
n02374451	horse
n04194289	ship
n04467665	truck

Table 1: Mapping ImageNet synsets to CIFAR-10 classes

The resulting download consists of 14,738 files with a total size of 1.3GB. Scaling the images to 224 x 224 pixels, the dimensions used by Szegedy, et al. decreased the size to a manageable 137MB.

After scaling, 100 images from each class were set aside to use for evaluating the trained model.

Training and validation sets were created in-situ using Tensorflow's image_dataset_from_directory function. The percentages were 80% for training, and 20% for validation.

5.2 Compute Resources

The networks created for this project were trained and tested using GPU and CPU resources for comparison purposes. The GPU is an NVIDIA Quadro P2000 GPU having 5GB of dedicated memory. The CPU is an Intel Core i9-10850K 3.6GHz CPU with 20 logical cores.

5.3 A Simplified GoogLeNet Network

Before constructing a complex model like GoogLeNet, it is a good idea to use a simple network as a baseline. This network is shown in Figure 2. This network is conceptually similar to GoogLeNet but without the complexity of the Inception modules. At it's core, the GoogLeNet architecture stacks multiple convolutional layers before down sampling through pooling. Stacking multiple layers helps networks capture progressively more complex features, improves generalization, and increases performance while keeping computational cost from getting out of hand. This is a common design pattern in many successful image classification networks [11].

The architecture of this network was informed by the following constraints:

- 10 image classes versus 1000 image classes in the complete ImageNet dataset.
- About 14.7K images instead of the over 1 million images in the complete ImageNet dataset.

These constraints were the reason why the filter depths range from 32 to 256.

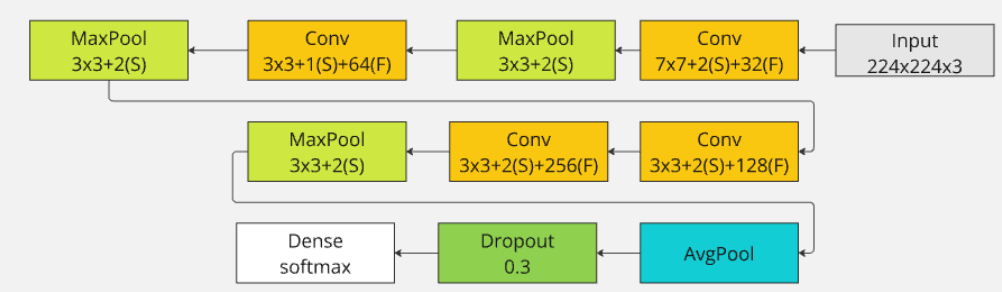

Figure 2: Simplified version of the GoogLeNet network

This simple model, when trained for 30 epochs, achieved a maximum accuracy of 57.2% at Epoch 23 as shown in Figure 3.

The validation accuracy and training accuracy track well, indicating a lack of overfitting. Applying data augmentation to the training set mitigates overfitting, especially with the smaller dataset.

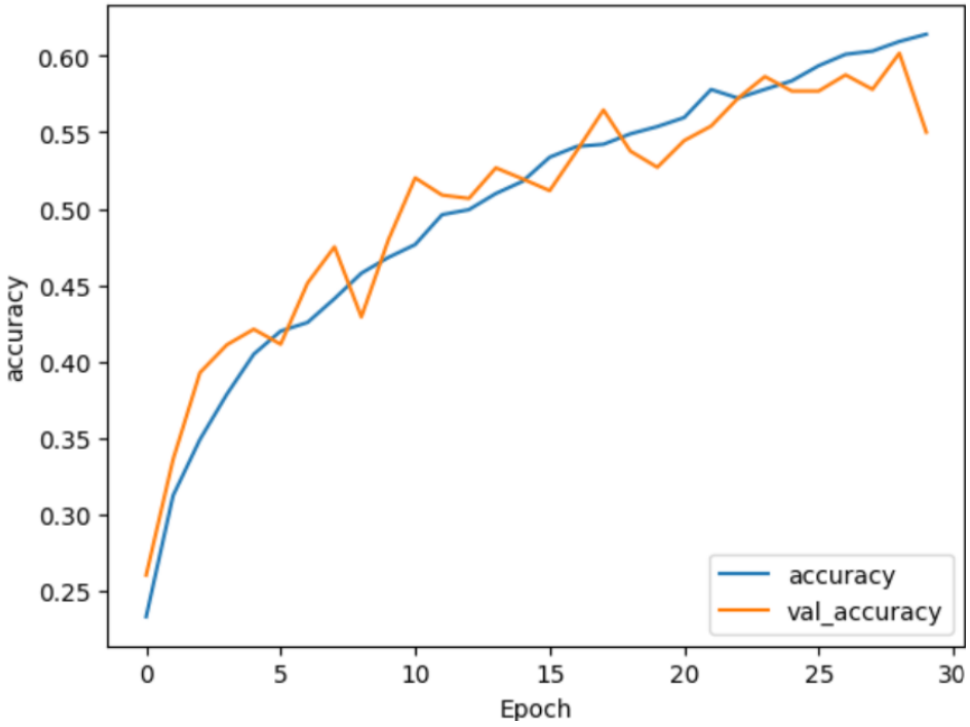

Figure 3: Accuracy of the simplified version of the GoogLeNet network

5.4 Replicating the GoogLeNet Network

The GoogLeNet architecture is explained in Table 1 of the Szegedy paper is reproduced in Figure 4. The network is 22 layers deep, incorporating branching and parallel convolutions via Inception modules. The introduction of Inception layers having parallel convolutional filters is the primary difference from typical sequential networks. The replication attempt, when trained for 30 epochs, achieved a maximum accuracy of 60.3%. This is quite improved over the simple model. The other impressive aspect is that it only took 14 epochs to arrive at the maximum accuracy as shown in Figure 5. The conclusion is that the architecture does provide marked improvements over the simpler, more naive model described in 5.3.

Nevertheless, this model is nowhere near the accuracy of the model in the

Szegedy paper. Their model achieved a top-5 error rate of only 6.7%. While the paper does not mention the accuracy, it is not a leap to assume that the accuracy was well above the accuracy in the replication attempt. We are optimistic, however, that our implementation is sound and that with a larger dataset, the accuracy can be improved and approach the accuracy achieved by Szegedy et. al. The network is graphically depicted in Figure 6

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 4: GoogLeNet architecture with Inception layers

5.5 Model Evaluation

Both networks were evaluated using two sets of images. The first is the 1,000 ImageNet images set aside prior to model training. The second set is sourced from the CIFAR-10 dataset [7]. The CIFAR-10 dataset contains 1,000 32x32 RGB images.

5.5.1 Performance On Unseen ImageNet Images

100 images from each class was set aside in order to evaluate the performance of the two models on unseen images. The accuracy of the two models is shown in 2.

The simpler model's accuracy of 58.1% on the test set is slightly better than the training accuracy of 57.2%. This indicates that the model generalizes

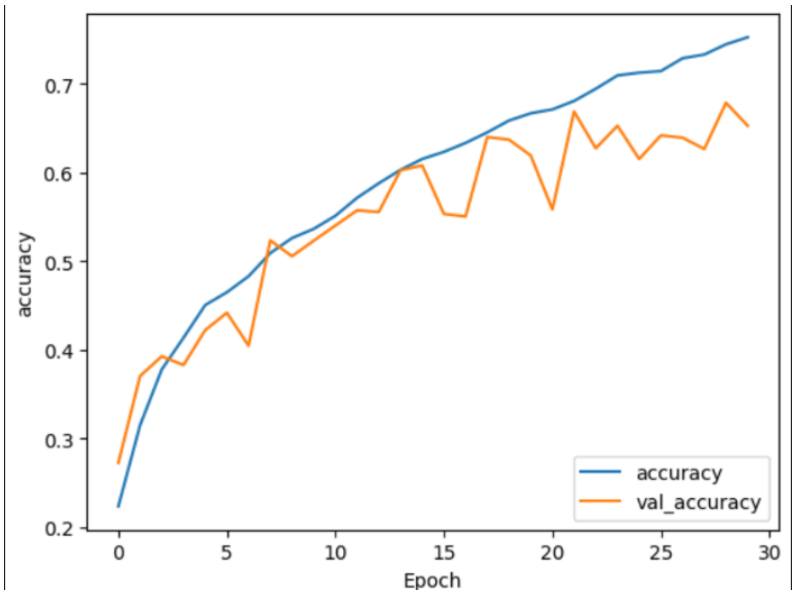

Figure 5: Accuracy of GoogLeNet network reproduction

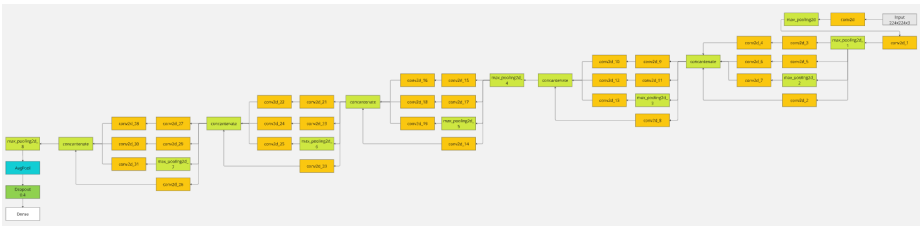

Figure 6: Replicated GoogLeNet network

well and did not merely memorize the training images. In machine learning, when a model performs well on unseen data compared to the training data, it suggests that the model has learned useful patterns that extend beyond the specific examples in the training set.

Furthermore, the relatively small difference in training accuracy and evaluation accuracy indicates that the model has managed to avoid overfitting. Overfitting typically results in high training accuracy but significantly lower evaluation accuracy. The fact that the model's test accuracy is slightly higher than its training accuracy can be attributed to augmentation done on the training set but not the evaluation set.

On the other hand, the 58.1% accuracy also indicates that there is room for improvement, suggesting that the model might benefit from more complexity or further tuning.

The GoogLeNet reproduction resulted in a 66.2% accuracy on unseen Ima-

Model	Accuracy
Simple	58.1%
GoogLeNet Reproduction	66.2%

Table 2: Models' performance on unseen ImageNet images

geNet images. The higher accuracy of this model can be attributed to the same reasons as the simpler model. In this case, additional complexity did provide a marked increase in accuracy. There is cause for optimism that a result closer to what was achieved by Szegedy, et al. could be matched with a larger training set.

5.5.2 Performance On CIFAR-10 Images

The CIFAR-10 dataset consists of 1,000 32x32 RGB images (100 images per class). The accuracy of the two models against the unseen CIFAR-10 images is shown in Table 3.

Model	Accuracy
Simple	21.4%
GoogLeNet Reproduction	22.6%

Table 3: Models' performance on unseen CIFAR-10 images

The low accuracy of both models could be for several reasons. First, the models were trained on images having a relatively high resolution compared to CIFAR-10's 32x32 images. When models are trained on larger images, the models learn to identify fine-grained details which can become indistinguishable when down scaled to much smaller dimensions [12].

Additionally, the convolutional filters in the models are likely too large for the small CIFAR-10 images, causing the network to miss out on important features. In a 32x32 image, for instance, using large filter sizes or pooling layers can result in excessive down sampling, leading to a significant loss of spatial information, which ultimately hampers the model's ability to classify the images accurately.

CIFAR-10, like ImageNet, contains complex image classes, which are often difficult to distinguish at such a low resolution. The limited information provided by 32x32 images presents additional challenges for models trained on high-resolution images, as they may rely on more detailed textures and edges, which are not as pronounced in low-resolution images [8].

Finally, Tensorflow up scales the images to 224 x 224 which surely results in a tremendous loss of fidelity. The fact that the accuracy is not lower is impressive when considering this loss of fidelity.

5.6 Challenges

The Inception architecture's key contribution is its ability to balance computational efficiency with accuracy. By incorporating 1x1 convolutions for dimensionality reduction and parallel convolutions in the Inception modules, GoogLeNet achieves leading performance without increased compute cost or model parameters.

In our reproduction, the model's performance on the ImageNet dataset was not close to the results reported in the original paper. This is not an indictment on the effectiveness of the architecture, however. Rather, the small dataset size could be a factor. Additionally, our interpretation of the Inception modules to code could have variations from the code used by Szegedy et al.

5.6.1 Computational Demand

One of the challenges we encountered was the high computational demand of training, even on the subset of ImageNet images we used. The training performance is summarized in Table 4. It is interesting that the training times for the two models were nearly identical when using a *single GPU* but differed by a factor of 1.7 when using *20 logical CPU's*. This drove home one of the reasons GPU's have become so popular. From the training times, it should be readily apparent why a subset of the ImageNet images were chosen and that training the whole dataset would take an inordinate amount of time with the compute resources we had access to. To further reduce the time, the learning rate was reduced to 0.0001, and batch size was reduced to 16. Prefetching was also implemented to improve performance [14]. ~~Although GoogLeNet is more efficient than many of its contemporaries, training still required GPU resources. We used data augmentation techniques, such as random cropping, random horizontal flipping, and random zoom to avoid overfitting [9]. We also needed to reduce the learning rate to 0.0001, reduced batch size to 16, and enabled memory growth [15] so as not to overwhelm the GPU. Prefetching was also implemented to improve performance [14]. The training performance is summarized in Table 4. From the training times, it should be readily apparent why a subset of the ImageNet images were chosen and that training the whole dataset would take an inordinate amount of time with the compute resources we had access to.~~

Model	Batch size	Epochs	Training time (minutes) GPU/CPU
Simple	16	30	35/42
GoogLeNet reproduction	16	30	37/71

Table 4: Training times

5.6.2 GPU Setup

The setup required to run Tensorflow code on a GPU on Windows is not trivial. Fortunately two good resources provided instructions on accomplishing this task [1] [10].

5.6.3 Overfitting

Overfitting was also an issue. We used data augmentation techniques, such as random cropping, random horizontal flipping, and random zoom to reduce overfitting [9].

5.7 Critical Assessment

The Inception architecture has had a significant impact on the field of deep learning, particularly in image classification. Its introduction marked a shift from simply increasing the size of models to incorporating more sophisticated designs that optimize the use of computational resources. This focus on efficiency has made the architecture suitable for deployment in resource-constrained environments, such as mobile devices.

The architecture has also influenced subsequent innovations, including more advanced object detection systems like Faster R-CNN and SSD, which build on the principles introduced by Inception. Additionally, later versions of the Inception architecture continue to push the boundaries of accuracy and efficiency in deep learning.

However, the architecture is not without its limitations. Despite its efficient design, training the network still requires significant compute power, especially for large datasets like ImageNet. Furthermore, the design of the Inception module, while effective, introduces additional complexity to the network architecture, making it challenging to implement and optimize.

5.8 Influence on Present Technology

Image recognition technology has become ubiquitous and is now readily accessible to the masses. For example, Google Photos can organize images based on their content [4], and one can take a picture of a food label written in a foreign language and translate it using Google Translate [5]. The success of neural networks can be largely attributed to the influence of Szegedy's work and its experiments. However, the original Szegedy experiments required substantial computing power, making them impractical for resource-constrained environments. Since 2015, there have been significant advancements in creating deep networks that can efficiently run on memory-limited devices, such as smartphones and automobiles. MobileNet, as described in [6], is a class of efficient models designed specifically for mobile applications. It builds upon key innovations from Inception, including multi-scale feature extraction, the use of 1x1 convolutions for dimensionality reduction, and the stacking of layers.

References

[1] Ali Abulhawa. Installing tensorflow 2.16 gpu on windows wsl2. https://medium.com/@ali.abulhawa/installing-tensorflow-2-16-gpu-on-windows-wsl2-df73ac3446c9, 2023. Accessed: 2024-10-10.

[2] conan7882. Googlenet inception - inception module implementation. https://github.com/conan7882/GoogLeNet-Inception/blob/master/src/models/inception_module.py, 2024. Accessed: 2024-9-10.

[3] DataCamp. Convolutional neural networks in python with tensorflow. https://www.datacamp.com/tutorial/cnn-tensorflow-python, 2024. Accessed: 2024-9-10.

[4] Google. Google photos organization updates - november 2023. https://blog.google/products/photos/google-photos-organization-updates-november-2023/, 2023. Accessed: 2024-10-10.

[5] Google. Translate text, images, or webpages. https://support.google.com/translate/answer/6142483?hl=en&co=GENIE.Platform=Desktop, 2024. Accessed: 2024-10-10.

[6] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In arXiv preprint arXiv:1704.04861, 2017.

[7] Yoongi Kim. Cifar-10-images: Python version of cifar-10 dataset (10 classes). https://github.com/YoongiKim/CIFAR-10-images, 2018. GitHub repository.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[9] Disha Sai. Multi-class image classification using the functional api of tensorflow. https://medium.com/@dishasai1997/multi-class-image-classification-using-the-functional-api-of-tensorflow-c95e656dab03, 2020. Accessed: 2024-09-22.

[10] Towards Data Science. Configuring jupyter notebook in windows subsystem linux (wsl2). https://towardsdatascience.com/configuring-jupyter-notebook-in-windows-subsystem-linux-wsl2-c757893e9d69, 2023. Accessed: 2024-10-10.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2015.

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

[14] TensorFlow. Better performance with the tf.data api. `https://www.tensorflow.org/guide/data_performance`, 2023. Accessed: 2024-09-27.

[15] TensorFlow. tf.config.experimental.set_memory_growth. `https://www.tensorflow.org/api_docs/python/tf/config/experimental/set_memory_growth`, 2023. Accessed: 2024-09-24.