Assignment 1: CNN

Comparing GoogleLeNet and DenseNet: Efficiency and Accuracy in Deep Learning

By

Ananya Krithika Thyagarajan (231871233)
Karthik Krishnamoorthy (23037687)

Table of Content

1 Introduction

GoogleLeNet and DenseNet are two innovative deep learning architectures that have garnered considerable interest in the world of computer vision. GoogleLeNet is a pioneering architecture in deep learning, renowned for its inception modules that enable efficient computing and deep networks. DenseNet is renowned for its distinctive method of connecting each layer to every other layer in a feed-forward manner, which encourages the reuse of features and leads to a substantial reduction in the number of parameters.

GoogleLeNet, which was introduced in the Inception architecture, employs a unique approach called network inside a network. This approach involves using numerous kernel sizes within the same layer to gather input at different scales. By employing this methodology, along with optimising the depth and width, GoogleLeNet is able to attain remarkable accuracy in image classification tasks while keeping the processing requirements very low.

DenseNet, proposed by Huang et al. (2016), extends this concept by creating a network with dense connections, where each layer is directly connected to all other layers. This design guarantees optimal transmission of information between layers, thus addressing the vanishing-gradient problem, enhancing feature propagation, and significantly decreasing the network's complexity. DenseNet has demonstrated remarkable efficacy across various domains, such as image classification, medical image analysis, and object identification, showcasing its versatility and efficiency.

1.1 Overview of GoogleLeNet

GoogLeNet is a significant advancement in convolutional neural networks (CNNs) that uses the Inception architecture to improve the ability to recognise images. This model effectively integrates various convolutional filter sizes into its Inception modules, enhancing both accuracy and processing efficiency.

The Inception Architecture:

The fundamental aspect of the Inception architecture is its capacity to dynamically choose the suitable size of the convolutional filter for every block of the network. GoogLeNet's capacity to adapt enables it to efficiently handle massive amounts of visual data, effectively managing performance and computational workload.

Computational Efficiency and Challenges:

Computational efficiency refers to the ability of a computer system or algorithm to perform tasks quickly and effectively, using few resources. It is a measure of how well a system can process and analyse data in a timely manner. However, there are various challenges that can hinder computational efficiency.

Although GoogLeNet is renowned for its effective utilisation of computational resources, its sophisticated design presents challenges in terms of implementation and optimisation. The model's significance in furthering deep learning technologies is emphasised by the delicate balance between efficiency and complexity.

1.2 Overview of DenseNet

DenseNet, also known as Densely Connected Convolutional Networks, is a notable progress in the structure of convolutional neural networks (CNNs). The design is original and has a distinctive connectivity pattern that enhances the flow of information and minimises the number of parameters. This makes it exceptionally efficient for a wide range of image processing jobs.

The DenseNet Architecture

DenseNet is distinguished by its feed-forward connectivity, where each layer is connected to every other layer. In DenseNet, the output of each layer is combined with the incoming feature mappings by concatenation, resulting in a composite function of learned features. This architectural design enables the use of deep supervision and feature reuse, which results in a significant increase in depth while using fewer parameters and addressing the issue of the vanishing gradient.

Computational Efficiency and Advantages

Computational efficiency refers to the ability of a computer system or algorithm to perform tasks quickly and effectively using minimal resources. It is a desirable characteristic that allows for faster and more efficient processing of data. The advantages of computational efficiency include improved performance, reduced energy consumption, and cost

DenseNet is highly efficient in terms of computation since it effectively manages parameters and enhances the flow of information within the network. The model has outstanding proficiency in tasks related to picture classification, segmentation, and recognition. The architecture of the model also enables substantial parameter reduction without compromising accuracy, a crucial accomplishment in the realm of deep learning.

Challenges and Impact

Although DenseNet offers numerous benefits, its dense interconnectedness results in higher memory and processing requirements, particularly in deeper networks. Advancements in memory optimisation approaches have facilitated improved training and use of DenseNet models. The contributions of DenseNet to deep learning, specifically in enhancing the efficiency of training and feature extraction, have resulted in its widespread adoption and significant influence in the field.

1.3 About the Data

The dataset we have used is the Kaggle Chest X-ray dataset, which is a comprehensive compilation of chest X-ray pictures frequently employed for training and assessing machine learning models in the field of medical diagnostics, particularly for the detection of illnesses such as pneumonia. The dataset has a substantial quantity of X-ray images that are classified into many categories, including normal, bacterial pneumonia, and viral pneumonia. These images are annotated for the purpose of supervised learning tasks.

The dataset is highly esteemed for its educational and research significance, serving as a substantial resource for the development and evaluation of deep learning algorithms, specifically convolutional neural networks (CNNs) such as GoogleLeNet and DenseNet. These networks have demonstrated exceptional achievements in tasks related to image classification.

2 Pre-processing of Data

2.1 Defining Dataset Paths

We first begin by establishing the file paths for the training, validation, and testing directories of the dataset.
In the code file, it is defined by:
dir_train: directory holding photos used for training data.
dir_val: directory holding validation images.
dir_test: directory that contains images used for testing purposes.

2.2 Function to Remove Corrupt Images

The **remove_corrupted_images(directory)** function traverses the designated directory, examines each image for corruption, and eliminates any photos that are corrupted. The process operates in the following manner:
Traverses through all subdirectories and files within the specified directory.
Attempts to open each image file and validate it using the img.verify() method.
If an image is determined to be corrupted, resulting in an IOError or SyntaxError, it is subsequently removed from the filesystem.
Retrieves and provides a list of deleted files to monitor the eliminated corrupted images.

2.3 Image Data Augmentation and Preprocessing

Data Augmentation for Training Data:

Data augmentation for training data involves utilising the “ImageDataGenerator” to specify the manner in which the training images will be modified in order to enhance generalisation and mitigate overfitting. The variables for augmentation comprise of the following:

1. **“rescale”:** Adjusts the pixel values of the image to fit into the range of [0, 1], which facilitates the training of the model.
2. The **"rotation_range"** parameter randomly rotates the photos within a specified range of degrees.
3. The parameters **"width_shift_range"** and **"height_shift_range"** introduce random horizontal and vertical shifts to the images.
4. **“shear_range”:** Implements random shearing transformations.
5. The **"zoom_range"** parameter randomly applies zooming effects to the photos.
6. The **"horizontal_flip"** function randomly mirrors the images along the horizontal axis.

Test and validation data:

Test and validation data do not undergo any data augmentation. The only operation performed is the rescaling of pixel values to a range of [0, 1]. No other forms of augmentation are used. This is done to ensure that the test and validation data accurately represent real-world data and are not artificially modified.

2.4 Loading Images from Directories

Uses the flow_from_directory method to load photos in batches from the provided directories for the purposes of training, validation, and testing.
dir_train, dir_val, dir_test: The file paths to the directories containing the data.
The target_size function resizes the photos to the supplied dimensions, which in this case are 150x150 pixels.
batch_size refers to the number of photos that will be produced by the generator in each batch. In this case, the batch size is set to 20.
class_mode: The kind of class labels to be returned is 'binary', indicating that there are two potential classes.

Ultimately, the code effectively identifies and communicates the quantity of images in each category, signifying that the dataset is prepared for the purpose of training a model. The preprocessing configuration described here is frequently used in deep learning workflows for image classification. In this case, ensuring the integrity of the data, applying augmentation techniques, and correctly loading the data are essential for successful model training.

Now , we will be training two CNN models and then comparing them.

3 GoogleLeNet

3.1 Network Structure and Hyperparameters

Base Model: InceptionV3

The function uses the InceptionV3 model as the base, taking advantage of its pre-trained weights from the ImageNet dataset. This option simplifies the process of extracting features from images by customising the model to eliminate the top classification layers, thereby enabling the generation of custom outputs.

Additional Layers and Classification

After applying the InceptionV3 model, a GlobalAveragePooling2D layer is used to compress the feature maps into a single vector for each map. This helps to decrease the complexity of the model and prevent overfitting. Afterwards, a densely connected layer consisting of 512 neurons utilises the Rectified Linear Unit (ReLU) activation function, which improves the network's ability to learn.

Output Layer and Optimisation

The last layer, utilising sigmoid activation, is specifically created for binary classification, generating a solitary output probability. The network employs the Adam optimizer and binary cross-entropy loss function, with accuracy as the major performance indicator.

3.2 Model Training and Evaluation

Training Performance

The Inception model underwent training for a total of 10 epochs, during which it consistently showed incremental enhancements. The initial accuracy of the model was high and continued to improve consistently, indicating its capacity to acquire knowledge from the pre-trained weights of ImageNet. Nevertheless, the validation accuracy exhibited periodic variations, indicating potential overfitting of the training data.

Evaluation Metrics

After evaluating the model using the test generator (test_Gen), the accuracy was found to be around 79.76%, with a loss of 26.4610. This ultimate performance parameter signifies the extent to which the model demonstrated its ability to apply to novel, unfamiliar data.

Interpretation of Results

The findings indicate that although the model has acquired the ability to accurately categorise images to a considerable extent, there is still potential for enhancement. This could be achieved by fine-tuning the model, implementing more robust regularisation techniques, or expanding the training dataset to enhance the model's ability to generalise.

3.3 Model Performance Metrics

Evaluation Metrics

After completing the training of the Inception model, we have computed essential metrics to assess its efficacy on unobserved data:

- **Accuracy:** Around 48.39%, representing the proportion of accurate forecasts out of the total number of predictions made.
- **Precision:** approximately 61.56%, indicating the percentage of accurate positive identifications.
- **Reminder:** Approximately 46.01% is the accuracy rate of correctly identifying genuine affirmative cases.

Interpretation of the Metrics

These metrics offer a thorough perspective on the model's ability to make accurate predictions. While accuracy is a broad metric, precision and recall are especially crucial in situations where the consequences of false positives and false negatives vary. The model has a higher level of precision compared to its recall, indicating a more cautious approach in predicting the positive class in order to minimise false positives. Nevertheless, the recollection suggests that there is potential for enhancing the ability to identify and include all pertinent occurrences.

3.4 Model Training and Inference Time Analysis

Training Time

The training length of the model for 10 epochs is roughly 2846.59 seconds, demonstrating the computational time needed to modify the weights and biases of the network using the whole training dataset.

Time taken to make an inference

After the training process, the time it takes for the model to make predictions on the test set, known as inference time, is approximately 11.53 seconds. This statistic is essential for assessing the model's feasibility in real-world scenarios where the speed of prediction may be of utmost importance.

Model Summary Insights

The model summary offers a comprehensive perspective on the architecture, encompassing the layers employed, output forms, and parameter count. The model's complexity is emphasised by its extensive parameter count, totaling over 68 million, with approximately 22

million of them being trainable. The intricacy of the model is a result of its vast feature extraction and learning capabilities. However, it also requires significant computer resources.

3.5 Hyperparameter Optimisation

Experimentation Overview

The performance of the model was evaluated using two different learning rates (0.001 and 0.0001) and two different optimizers (Adam and SGD). The objective of this experiment is to determine the most effective pairing of learning rate and optimizer for the Inception model in binary classification problems.

Summary of Results

Using a learning rate of 0.001 and the Adam optimizer, the model demonstrated significant accuracy. However, there is some evidence of overfitting, as the validation accuracy is lower than the training accuracy.

Adopting the SGD optimizer with an identical learning rate led to a marginal decline in training accuracy but a more stable validation accuracy, indicating improved generalisation.

By decreasing the learning rate to 0.0001 and employing the Adam optimisation algorithm, a significant decrease in training accuracy was seen, most likely attributed to the slower convergence of the model.

The stochastic gradient descent (SGD) optimizer, when used with a decreased learning rate, exhibited decreased training accuracy and a marginal enhancement in validation accuracy.

Interpretation

Using the Adam optimizer with a larger learning rate results in faster convergence, but it also increases the risk of overfitting. This is evident from the higher training accuracies but lower validation accuracies. Conversely, the SGD optimizer enhances overall performance but takes longer to reach optimal results, as evidenced by decreased accuracy but increased validation consistency. Optimal model performance relies on finding the right balance between the learning rate and the type of optimizer used.

3.6 Model Training Visualisation

Trends in Accuracy

The accuracy plots depict the model's capacity to accurately categorise the training and validation data throughout different epochs. The training accuracy is consistently high, suggesting that the model is highly skilled at acquiring knowledge from the provided data. Nevertheless, the validation accuracy exhibits substantial fluctuations, indicating that the model's ability to perform effectively on unseen data may be limited.

Trends in Losses

The loss graphs provide insight into the model's optimisation of its internal parameters. The training loss exhibits a consistent and gradual decline, as anticipated with the model's learning process. In contrast, the validation loss exhibits sudden increases, indicating the possibility of overfitting or the model's susceptibility to the particular validation set employed.

Observations

The presence of a persistent disparity in the accuracy of the model on the training and validation data, as well as an inverse correlation observed in the loss plot, suggests that the model is able to learn from the training data but is not able to consistently perform well on the validation data. To resolve this discrepancy, one can utilise strategies such as data augmentation, dropout, or regularisation to enhance the model's capacity to generalise.

4 DenseNet

4.1 Network Structure and Hyperparameters

Base Model: DenseNet121

The code employs DenseNet121 as the primary model, which is initialised with pre-trained ImageNet weights to exploit its robust feature extraction capabilities. To facilitate the incorporation of bespoke layers tailored for specific tasks, the uppermost categorization levels are omitted.

Additional Layers and Classification

After the DenseNet121 model, a GlobalAveragePooling2D layer is used to combine the feature maps into a single vector per map. This helps simplify the model and decreases the likelihood of overfitting. A layer with a high concentration of 256 neurons and ReLU activation is added, which increases the network's ability to understand intricate patterns.

Output Layer and Optimisation

The model is finalised by adding a dense output layer with sigmoid activation, which produces a probability output for binary classification. The network utilises the Adam optimizer with a learning rate of 0.001 and employs binary cross entropy as the loss function, with a focus on accuracy as the primary performance metric.

Layer Freezing and Model Compilation

The base DenseNet121 layers are immobilised to preserve their pre-trained characteristics throughout the early training stages, inhibiting any modification to their weights. Transfer learning relies on this step to ensure that the new layers are able to learn effectively. Next, the model is compiled using the designated optimizer, loss function, and accuracy metric, which prepares it for both training and evaluation.

4.2 Model Training and Evaluation

Training Performance

The DenseNet model underwent training for 25 epochs, demonstrating progressive enhancements in accuracy throughout the process. The initial accuracy was 65.42% and varied throughout the training process, suggesting that the model was acquiring knowledge and adjusting its performance based on the pre-trained weights from ImageNet. Although there were some initial variations, the model achieved a training accuracy of 94.16%, indicating its promising learning potential. Nevertheless, the validation accuracy exhibited fluctuations, indicating that the model may be excessively conforming to the training data.

Evaluation Metrics
Performing model evaluation using the test set is essential for assessing the model's ability to generalise to unseen data. This step will evaluate the ultimate precision and error, offering a valuable understanding of the model's effectiveness in a real-life situation.

Interpretation of Results
The fluctuating validation accuracy suggests that the model's predictions on the validation set lack consistency. One way to tackle this issue is by utilising methods like data augmentation, dropout, or regularisation to enhance generalisation. The training trend also indicates the possibility of adjusting hyperparameters or implementing early stopping to avoid overfitting.

Performance Analysis
A graph is generated to display the accuracies of the training and validation sets as a function of the number of epochs. These plots are crucial for illustrating the learning process and identifying discrepancies in performance between training and validation. Such discrepancies may indicate the presence of overfitting or underfitting. The plots aid in assessing the need for additional model tweaks prior to implementing the model for practical use.

4.3 Model Performance Metrics

Evaluation Metrics
The DenseNet model was evaluated on the test set and produced the following important metrics:

- Accuracy: The model achieved an approximate accuracy of 81.91%, which is the proportion of correct predictions made by the model out of all the predictions.
- Loss: The statistic recorded at 0.4419 indicates the model's average error and serves as a performance indicator, with lower values indicating greater performance.

Additional metrics for comprehensive analysis
Furthermore, apart from evaluating the model's accuracy and loss, precision and recall were also considered.
Achieved a precision rate of 62.80%, demonstrating the model's efficacy in accurately categorising an event as positive.
The model achieved a recall rate of 79.23%, indicating its capacity to correctly identify all pertinent events in the dataset.

Interpretation of the Metrics
The combination of these measures provides a comprehensive comprehension of the model's performance:

- Accuracy signifies a superior level of overall performance.
- The accuracy indicates that the model correctly identifies an X-ray as suggestive of a condition around 62.80% of the time.

- The high recall indicates that the model is relatively dependable in identifying the positive class, but it may also result in a greater number of false positives.
- The loss value indicates that there is still residual error in the model predictions, which could potentially be minimised with additional fine-tuning.

These insights can be used to improve the model by making adjustments to thresholds, continuing training, increasing the dataset, or applying new strategies to balance precision and recall.

Consideration of Training Time

The model's training duration was observed to be approximately 16.94 seconds, a crucial aspect to consider when evaluating the efficiency and scalability of training such models in real-world scenarios.

4.4 Hyperparameter Optimisation

Experimentation Overview

The purpose was to optimise the hyperparameters of a DenseNet121 model, specifically investigating the effects of varying numbers of dense layer units and learning rates. The Optuna framework was utilised to construct and compile the model, incorporating several dense units and learning rates to identify the optimal combination for binary classification tasks.

Summary of Results

The model's performance exhibited variability across numerous trials due to variations in hyperparameters. It was concluded that the most effective setup consists of 256 dense layer units with a learning rate of roughly 0.0033. This configuration attained the utmost validation accuracy, indicating that it was the most efficient in extrapolating from the training data.

Interpretation

It emphasises the significance of hyperparameter adjustment in machine learning models. The choice of units in the dense layer and the learning rate have a direct impact on the model's capacity to learn and generalise. The optimal hyperparameters achieved a harmonious trade-off between learning complexity and convergence speed, enabling the model to exhibit excellent performance on the validation data.

The Optuna framework expedited the process by effectively exploring the hyperparameter space and assessing the performance of each trial. The identified ideal hyperparameters will be used to build the final DenseNet121 model for subsequent training and evaluation tasks.

4.5 Model Training Visualisation

Trends in Accuracy

Upon analysing the accuracy trends seen during the model's training process, it becomes evident that the training accuracy remains stable, suggesting that the model is adept at acquiring knowledge from the training dataset. Nevertheless, notable variations in validation accuracy indicate difficulties in the model's ability to apply to unfamiliar data.

Trends in Losses

The training loss trends exhibit a progressive decline, in accordance with the anticipated behaviour as the model optimises its parameters to more accurately match the training data. On the other hand, the validation loss exhibits fluctuations with significant peaks, which could indicate overfitting or a susceptibility to the specific validation dataset employed.

Significant observations

The continual discrepancy between the accuracy of the model on the training and validation data, as well as the conflicting patterns in loss, indicates that the model performs well on the training data but shows inconsistent performance on the validation data. In order to address this issue, techniques such as data augmentation, the inclusion of dropout layers, or additional regularisation methods could be utilised to improve generalisation.

Model Optimisation Analysis

After modifying the hyperparameters, the model demonstrates a higher validation accuracy and a more consistent trend in validation loss, indicating greater generalisation ability. However, intermittent increases in validation loss suggest the possibility of overfitting, which can be mitigated by adjusting the model's complexity or the training procedure.

Hyperparameter-Tuned Model Performance

Once the tuned model is used to make predictions on the test set, we proceed to evaluate its performance.

- Tuned Model Accuracy: The accuracy after tuning is roughly 87.79%, which is the percentage of true predictions provided by the model.
- Optimised Model: Loss: The tuning process results in a loss of approximately 0.2943, which represents the model's prediction error. It is desirable to have smaller values for loss.

Comparative Analysis

The enhanced accuracy and diminished loss of the tuned model, in comparison to the initial model, indicate the efficacy of the hyperparameter optimisation. The optimisation approach yielded a refined model that exhibits enhanced generalisation capabilities towards unseen data. However, the precision and recall metrics suggest that further enhancements can still be made.

The optimised model attained the subsequent metrics:

- The accuracy of the prediction is approximately 54.49%, indicating the percentage of right predictions.
- Accuracy: approximately 62.38%, representing the precision of optimistic forecasts.
- The recall rate is approximately 68.46%, which indicates the model's capacity to correctly identify all relevant events.
- These indicators offer valuable information about the customised model's ability to make accurate predictions, showcasing its strengths and identifying areas for further improvement. The optimisation of the model has modified the balance between precision and recall, perhaps sacrificing raw accuracy but resulting in improved capability to accurately detect affirmative cases.

5: Comparative Analysis of GoogleLeNet (InceptionV3) and DenseNet

5.1 Model Training and Evaluation

Training Performance:

The GoogleLeNet model showed consistent enhancements during the course of training; however, there were variations in the validation accuracy, suggesting the possibility of overfitting. On the other hand, after optimising the hyperparameters, DenseNet showed consistent enhancements and a favourable capacity for learning, with increased reliability in validation accuracy.

Evaluation Metrics:

GoogleLeNet obtained an accuracy of approximately 79.76% on the test set, indicating a satisfactory level of generalisation. After tuning, DenseNet achieved a greater accuracy of around 87.79%, demonstrating improved predictive performance due to optimised hyperparameters.

5.2 Hyperparameter Impact and Optimisation

GoogleLeNet Optimisation:

Experiments conducted on GoogleLeNet revealed that using a higher learning rate in conjunction with the Adam optimizer resulted in quicker convergence. However, this approach also posed a greater danger of overfitting. The SGD optimizer, when used with the same learning rate, exhibited a higher level of consistency in validation accuracy, indicating superior generalisation.

DenseNet Optimisation:

Through Optuna, hyperparameter optimisation for DenseNet demonstrated that specific dense units and learning rates significantly enhanced the model's performance. An ideal learning rate of around 0.0033 was determined, along with 256 dense units.

5.3 Model Comparison and Key Findings

Accuracy and Loss Trends:

The validation accuracy and loss of GoogleLeNet showed greater variations, while DenseNet showed more stability after tuning. This indicates that DenseNet may be more resistant to overfitting after the hyperparameters are optimised.

Generalisation:

DenseNet's architecture promotes the reuse of features, resulting in improved generalisation. This is evident in the more consistent validation accuracy and loss trends compared to GoogleLeNet.

Computational Efficiency:

Although DenseNet is recognised for its parameter count efficiency, GoogleLeNet's inception modules provide a competitive trade-off between accuracy and computational burden.

Precision and recall:

Both models demonstrate disparities in precision and recall, with DenseNet displaying enhancement following hyperparameter optimisation. Nevertheless, there exists a compromise between accuracy and completeness, and the selection of a model may vary based on the specific requirements of the application.

Conclusion

Both GoogleLeNet and DenseNet are robust frameworks for picture categorization. DenseNet, especially when combined with hyperparameter adjustment, appears to have an advantage in terms of both performance stability and accuracy. However, the task's specific requirements should ultimately guide the choice between the two options, taking into account factors like the complexity of the dataset, the availability of computational resources, and the importance of training time relative to inference time. Optimising hyperparameters is essential for maximising the potential of a model, as demonstrated by the enhanced performance of DenseNet after tuning.

Reference

- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (n.d.). GoogLeNet. Papers with Code. Retrieved from https://paperswithcode.com/method/googlenet
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *CVPR*. Retrieved from https://ieeexplore.ieee.org/document/8099726
- Zhou, T., Ye, X., Lu, H., Zheng, X., Qiu, S., & Liu, Y. (2022). Dense Convolutional Network and Its Application in Medical Image Analysis. *BioMed Research International*. Retrieved from https://www.hindawi.com/journals/bmri/2022/3407389/
- Pleiss, G., Chen, D., Huang, G., Li, T., Van Der Maaten, L., & Weinberger, K. Q. (2017). Memory-Efficient Implementation of DenseNets. *ArXiv*. Retrieved from https://arxiv.org/abs/1707.06990
- Chen, Z., & Liu, G. (2019). DenseNet+Inception and Its Application for Electronic Transaction Fraud Detection. *IEEE 21st International Conference on High Performance Computing and Communications*. Retrieved from https://ieeexplore.ieee.org/document/8846962