

Food Classification Using Deep Learning

Farrukh Nizam Arain
Department of Computing and
Mathematics
Manchester Metropolitan University
Manchester, United Kingdom
12345678@stu.mmu.ac.uk

Abstract—A growing number of people are becoming conscious about the food that they consume or produce. They want to estimate the nutritional content of food to manage their health and dietary goals, search for suitable recipe recommendations based on food images, or even choose restaurants based on calorie count. The government also wants to maintain food safety standards through inspection of the quality of imported and domestically produced food. In recent times, deep learning has shown promise in developing advanced techniques to distinguish food with impressive accuracy. This research aims to propose a food classification system through three different CNN-based deep learning algorithms, which are ResNet34, DenseNet121, and VGG-16. In this study, a food classification dataset is used, which is grouped into 11 major categories. According to the results, all the algorithms had excellent accuracy, with DenseNet121 being in the top spot amongst all the algorithms when the accuracy and evaluation scores were compared.

Keywords—Deep Learning, Food Classification, CNN, ResNet34, DenseNet121, VGG-16

I. INTRODUCTION

Around the world, individuals are becoming more interested in knowing the nutritional content and benefits of different foods because it is essential for health, weight management, disease prevention, and allergies. Even if one does not have diseases or allergies to be careful of, by understanding which foods to consume and which to avoid, individuals can make informed choices, meet their dietary needs, and even be concerned about environmental impacts on food choices. According to the 2022 Food and Health Survey by the International Food Information Council (IFIC), consumers are increasingly focusing on all the aspects of food choices mentioned earlier to not only make their own lives better but also to contribute to the betterment of the global environment [1].

With the advancement of computer software and hardware, food classification through deep learning helps in automatically analysing and categorizing food into different classes. Different food classification algorithms like Convolutional Neural Networks (CNN), a powerful deep learning architecture, play a vital role in recognizing complex patterns in images and classifying them into different classes so that consumers can not only modify their dietary habits but also make use of these techniques to improve their and other people's lives [2].

The aim of this research is to build a prototype food classification system that analyses thousands of food images using deep learning. For the accomplishment of this research, a sample food classification database is used which contains thousands of images of multiple classes. Three CNN-based algorithms, which are ResNet34, DenseNet121, and VGG16, were trained on the dataset, as these are known to perform well in differentiating images into different classes. The input to the algorithm is an image, which is processed by the

algorithms to output the class of that image based on the patterns learned earlier. The performance of the algorithms is evaluated via multiple metrics to ensure that the most suitable algorithm is chosen from all aspects known.

The remainder of the research paper is structured as follows: Section II provides the link to the project. Section III provides a summary of a thorough study of related research papers. Section IV describes the dataset used in the research paper. Section V describes the methodologies of all the learning algorithms. Hyperparameters, experimental results, and evaluation results of each algorithm are discussed in Section VI. Section VII compares the results through a number of evaluation metrics. Research limitation faced are discussed in Section VIII. Lastly, Section IX is about the conclusion of the research, selecting the best performing algorithm, and mentioning future recommendations.

II. PROJECT PROTOTYPE

The complete project prototype for implementing this research is available on the GitHub link provided below.

<https://github.com/farrukhna/deep-learning>

Additionally, an example of the primary project notebook running on Kaggle is also accessible at:

<https://www.kaggle.com/code/farrukhnizamrain/food-classification-final>

III. RELATED WORK

One study introduced DeepFood, a CNN-based system for accurately recognizing various food items to assist in dietary assessment [3]. While achieving high accuracy, it needed substantial training data. It resembled this research in CNN-based food image classification, though with a focus on dietary assessment rather than general food recognition. Another study compared multiple deep learning models for early detection of rice diseases [4]. It provided a comprehensive analysis, yet limited its scope to agricultural disease detection, distinct from general food image classification. While both studies implemented various deep learning models, the focus of this study remains on food image classification rather than agricultural disease detection. Moreover, one study explored food ingredient classification using transfer learning techniques, primarily employing CNN architectures like VGG-16 and ResNet-50 [5]. While it provided a thorough comparison and utilized transfer learning to boost accuracy, it had limitations in customization and immense computational demands. In contrast, although this research used similar architectures, this study focuses on general food image classification without emphasizing transfer learning techniques. One research developed an automated food image classification system using CNNs, emphasizing its application in diet monitoring within the

health and medical domains [6]. Although it achieved high accuracy through deep learning techniques, it highlighted the potential risk of overfitting on limited datasets. While both studies utilized CNNs for food classification, this research expands its scope by integrating ResNet34, DenseNet121, and VGG-16 models, offering a comprehensive comparative analysis beyond conventional CNN architectures. Another study explored improving food recognition accuracy by utilizing deep convolutional features extracted from CNNs, employing pre-trained models and fine-tuning them for the task [7]. While this approach enhanced accuracy, it also posed challenges in computational resources and time consumption. In contrast to that paper, this research expands the scope by including ResNet34 and DenseNet121 alongside traditional CNN architectures for food image recognition.

IV. DATASET DESCRIPTION AND ANALYSIS

The project uses the Food-11 image dataset [8], sourced from Kaggle, comprising of 16,643 images categorized into 11 distinct food classes: Bread, Dairy product, Dessert, Egg, Fried food, Meat, Noodles/Pasta, Rice, Seafood, Soup, and Vegetable/Fruit. These images are already cleaned and are evenly distributed across the classes. The dataset is divided into training, validation, and evaluation subsets, with 9866, 3430, and 3347 images respectively, ensuring a balanced representation of classes in each subset.

To enhance model robustness and generalization, data augmentation techniques are applied during preprocessing. For the training data, the images are first resized to a standard size of 224x224 pixels. Then, random rotations up to 20 degrees are applied and random horizontal flips as well to augment the data and make the model more robust to variations in the input. Colour jittering is also applied to randomly change the brightness, contrast, and saturation of the images. After these augmentations, the images are converted to PyTorch tensors and then normalized using the provided mean and standard deviation values. Finally, random erasing is applied, which randomly selects a rectangular region in the image and erases its pixels with random values, providing a form of cutout augmentation. For the validation and test data, the images are resized, converted to tensors, and then normalized. The reason for not augmenting the validation and evaluation dataset is to evaluate the models on the original data, not on artificially created variations. A sample of the images from the dataset (Figure 1) ensures that the initial setup is ready to be processed by the deep learning algorithms.



Figure 1: Sample images from the dataset

V. METHODOLOGY

Following is the methodology and architectural description of the models used for the project:

a) ResNet34: ResNet34, part of the Residual Network (ResNet) family introduced by Kaiming He et al. in 2015 [9], is a convolutional neural network designed to mitigate the vanishing gradient problem in deep networks using residual connections, making it suitable for training very deep models effectively. The architecture (Figure 2) begins with a 7x7 convolution layer with 64 kernels, a stride of 2, and is

followed by MaxPooling. It includes four residual blocks with configurations of 3, 4, 6, and 3, respectively, and each block has channels of 64, 128, 256, and 512. Only 3x3 kernels are used within these blocks, and except for the first block, each block starts with a 3x3 kernel with a stride of 2. Dotted lines represent skip connections, which can be added directly when input and output dimensions match; otherwise, a 1x1 convolution with a stride of 2 is used to match dimensions. For ResNet34, the residual block configuration is 3, 4, 6, 3, while ResNet18 has a 2, 2, 2, 2 configuration. Other versions include ResNet Bottleneck (R50, R101, R152), ResNet V3, and ResNeXt.

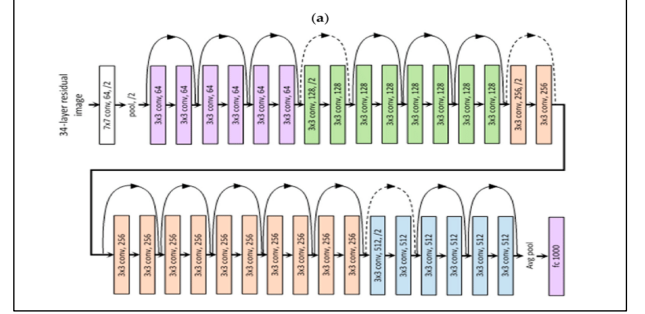


Figure 2: ResNet-34 architecture

b) DenseNet121: DenseNet121 is a deep convolutional neural network known for its efficient architecture that promotes feature reuse through dense connectivity, where each layer receives inputs from all preceding layers [10]. The architecture (Figure 3) begins with a 7x7 convolution layer (C1) with 64 kernels, a stride of 2, and is followed by a 3x3 MaxPooling layer. It includes four dense blocks (D1, D2, D3, D4) with 6, 12, 24, and 16 layers, respectively. Between these dense blocks, there are three transition layers (T1, T2, T3), each consisting of a 1x1 convolution followed by a 2x2 average pooling layer with a stride of 2. Each layer within the dense blocks uses 3x3 convolutions. Bottleneck layers, which are 1x1 convolutions, are used before the 3x3 convolutions to reduce the number of input feature maps, enhancing computational efficiency. Skip connections are used extensively, where each layer receives additional inputs from all preceding layers within the same dense block, promoting feature reuse and improving gradient flow. The architecture concludes with a global average pooling layer and a fully connected layer for classification.

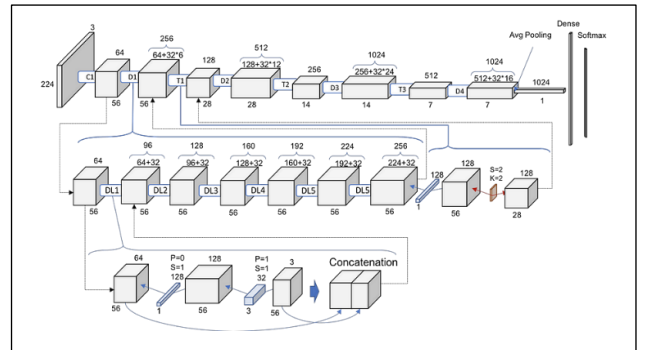


Figure 3: DenseNet121 architecture

c) VGG-16: VGG-16 is a convolutional neural network architecture introduced by the Visual Geometry Group (VGG) at the University of Oxford, known for its simplicity and depth, achieving excellent performance in image classification tasks [11]. Its architecture (Figure 4) consists of

16 layers: 13 convolutional layers and 3 fully connected layers. The network starts with two convolutional layers with 64 filters each, followed by a max pooling layer, then two convolutional layers with 128 filters each, followed by another max pooling layer. This pattern continues with three convolutional layers with 256 filters each, followed by a max pooling layer, and then three convolutional layers with 512 filters each, followed by a max pooling layer. Finally, it has three convolutional layers with 512 filters each again, followed by a max pooling layer. The convolutional layers use small receptive fields (3x3) and ReLU activation functions. The final part of the network consists of three fully connected layers: the first two have 4096 units each, and the last one has 1000 units for the 1000-class classification task, all with ReLU activations, ending with a softmax layer for output. This deep architecture, with its repeated patterns of convolution and pooling, enables VGG-16 to capture intricate features and achieve high accuracy in visual recognition tasks.

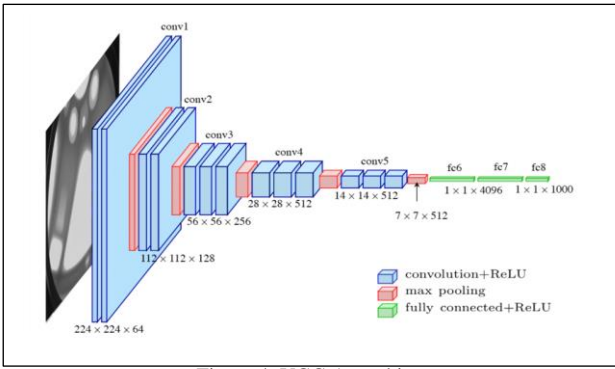


Figure 4: VGG-16 architecture

VI. EXPERIMENTATION RESULTS

The performances of the models and their individual analysis are described below:

a) *ResNet34*: The training strategy involves 15 epochs, with early stopping if there's no improvement in validation accuracy for 3 consecutive epochs. The model is trained on a GPU if available, using Stochastic Gradient Descent (SGD) with a learning rate of 0.001 and momentum of 0.9. The learning rate is halved every 5 epochs. The loss function used is Cross-Entropy Loss. The model's performance is evaluated using accuracy, precision, recall, F1 score, and AUC-ROC. The best model is saved based on the highest validation accuracy. The results show that the model achieved a test accuracy of approximately 92.68%, precision of 92.68%, recall of 92.68%, F1 score of 92.66%, and AUC-ROC of 95.87%. These metrics indicate that the model has a good balance between precision and recall, and a high AUC-ROC suggests that the model has a high discriminative power. The early stopping mechanism, along with the relatively high performance on the validation set, suggests that the model has not overfitted to the training set. Additionally, the code includes a function to display several of the predicted images along with their predicted and true labels to reaffirm the excellent performance of the algorithm of the dataset. In addition to that, it presents with a couple of line plots regarding the train and validation loss, and also for train and validation accuracy to show in which EPOCHS the algorithm performed the best. Confusion matrix is also generated of the prediction values (Figure 5) to show that the algorithm predicted correct labels for most of the images.

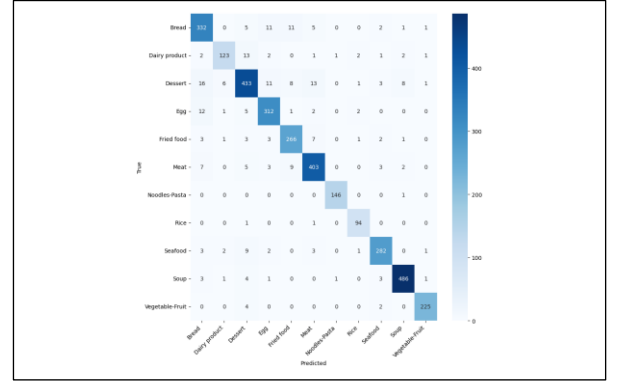


Figure 5: Confusion matrix of predictions by ResNet34

b) *DenseNet121*: The model was pretrained, and the final layer was replaced to match the number of classes in the dataset. Training lasted for 15 epochs with a learning rate of 0.001 and momentum of 0.9, utilizing Stochastic Gradient Descent (SGD) as the optimizer and Cross Entropy Loss as the loss function. The model was trained on a GPU if available; otherwise, it was trained on a CPU. Training halted if the validation accuracy did not improve for 3 consecutive epochs. The model's performance was evaluated using accuracy, precision, recall, F1 score, AUC-ROC, and confusion matrix (Figure 6). The best model achieved a validation accuracy of approximately 91.37% and was saved. Subsequently, the model was tested and attained an accuracy of around 93.07%, precision of 93.10%, recall of 93.06%, F1 score of 93.06%, and AUC-ROC of 96.11%. These results suggest that the model performed well on the test set, indicating that it did not overfit the training set. The model's high precision and recall imply its ability to correctly classify most of the food items, while the high AUC-ROC indicates its effective distinction between different classes.

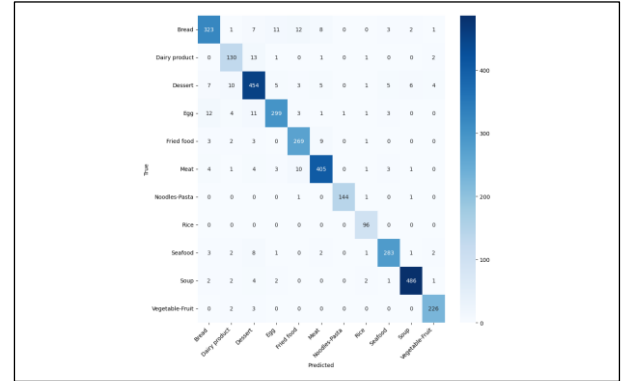


Figure 6: Confusion matrix of predictions by DenseNet121

c) *VGG-16*: The food classification task utilized the VGG-16 model, which was pre-trained and fine-tuned for 11 classes. Training spanned 15 epochs, employing a learning rate of 0.001 and momentum of 0.9 for the SGD optimizer. If available, training took place on a GPU, and the process ceased if there was no improvement in validation accuracy for 3 consecutive epochs, aiming to prevent overfitting and unnecessary computation. Evaluation of the model's performance encompassed metrics such as accuracy, precision, recall, F1-score, and AUC-ROC. The best model was preserved and subsequently evaluated on the test set, revealing a test accuracy, precision, recall, and F1-score all at

approximately 92%, with an AUC-ROC of 95.45%. These metrics shows a creditable balance between precision and recall, with the high AUC-ROC indicating strong separability. Notably, the model appears not to overfit the training set, given the proximity of validation accuracy to training accuracy. Moreover, the early stopping mechanism ensured stoppage of training when performance ceased to improve on the validation set. Confusion matrix of the prediction (Figure 7) values also shows that the algorithm detected the classification of most of the images successfully.

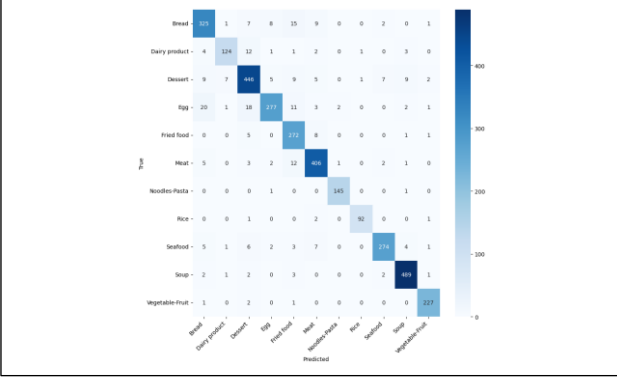


Figure 7: Confusion matrix of predictions by VGG-16

VII. COMPARISON OF RESULTS AND DISCUSSION

In a side-by-side comparison of results illustrated in Table 1, DenseNet121 achieved the highest scores across all metrics, with an accuracy of 93.07%, precision of 93.10%, recall of 93.06%, F1-score of 93.06%, and AUC-ROC of 96.11%. In comparison, VGG16 achieved an accuracy of 91.94%, precision of 92.06%, recall of 91.93%, F1-score of 91.91%, and AUC-ROC of 95.45%. ResNet34, on the other hand, achieved an accuracy of 92.68%, precision of 92.68%, recall of 92.68%, F1-score of 92.66%, and AUC-ROC of 95.87%. These results clearly indicate that DenseNet121 outperformed VGG16 and ResNet34 in classifying the test images accurately.

Model Name	Accuracy	Precision	Recall	F1-Score	AUC
ResNet34	92%	92%	92%	92%	95%
DenseNet121	93%	93%	93%	93%	96%
VGG-16	91%	92%	91%	91%	95%

TABLE I: Prediction results of the algorithms

The difference in performance among the models can be attributed to their architecture and the way they process information. DenseNet121 has a unique structure where each layer is directly connected to every other layer in a feed-forward fashion, which improves the flow of information and gradients throughout the network, making it easier to train. This could be why it outperformed the other models. For instance, if the DenseNet121 model achieved an accuracy of approximately 93%, while the VGG16 and ResNet34 models achieved accuracies of 91% and 92% respectively, this 1-2% difference could be due to the superior architecture of DenseNet121. On the other hand, VGG16 and ResNet34, despite being powerful models, might not have been the best fit for this specific task. The depth of VGG16 could have led to overfitting, while the residual connections in ResNet34,

although designed to solve the vanishing gradient problem, might not have been as effective as the dense connections in DenseNet121 for this particular problem.

VIII. RESEARCH LIMITATIONS

Due to the local machine not being powerful enough, a significant limitation of this research was due to the restricted access to computational resources. The experiments were initially conducted on Google Colab, a cloud-based development environment, using a single notebook. However, without access to a dedicated GPU and with limited available RAM, these issues necessitated a switch to Kaggle's cloud-based development environment. Although Kaggle provided two T4 GPUs, usage was limited to 30 hours on the GPU, which was insufficient for such a large dataset. As a result, the notebook had to be executed multiple times with a small number of epochs, and only the most reliable results are presented in this paper.

IX. CONCLUSION AND FUTURE RECOMMENDATIONS

In conclusion, the evaluation of three deep learning models—ResNet34, DenseNet121, and VGG16—on the test set revealed that while all models performed well, the DenseNet121 model outperformed the others. ResNet34 achieved an accuracy of 92.68%, precision of 92.68%, recall of 92.68%, F1-score of 92.66%, and AUC-ROC of 95.87%, demonstrating good overall performance with room for improvement. DenseNet121, with its advanced architecture connecting each layer to every other layer in a feed-forward manner, achieved the highest scores: 93.07% accuracy, 93.10% precision, 93.06% recall, 93.06% F1-score, and 96.11% AUC-ROC, indicating superior classification effectiveness. The VGG16 model, although it also had respectable accuracy scores, lagged behind with scores of 91.94% accuracy, 92.06% precision, 91.93% recall, 91.91% F1-score, and 95.45% AUC-ROC, suggesting it was less effective at capturing complex features. These results suggest that the DenseNet121 architecture may be the most suitable for this image classification task, though exploring different models and hyperparameters could further optimize performance.

Given access to more time, team members, and computational resources, several avenues could be explored to enhance this project. More experimentation with data augmentation techniques could be employed to increase the diversity of the training data, potentially improving the models' ability to generalize. Additionally, a more extensive hyperparameter testing could be conducted to find the optimal settings for each model, thereby enhancing their performance. With access to a much more powerful local machine without restrictions on computational power, experiments could be conducted more efficiently and on a larger scale, allowing for deeper exploration of various methodologies and models.

ACKNOWLEDGEMENT

I extend my sincere gratitude to Module (Unit) Leader Dr. Xinqi Fan, Lecturer Dr. Nashid Alam, and Teaching Assistant Christian Jordan for their invaluable guidance and instruction in laying the groundwork for this research. Their expertise and support have been instrumental in shaping the foundations of this work.

REFERENCES

- [1] "2022 Food and Health Survey Report," May 2022. [Online]. Available: <https://foodinsight.org/wp-content/uploads/2022/05/IFIC-2022-Food-and-Health-Survey-Report.pdf>
- [2] R. Volety, "ML Guide to Train Food Recognition and Classification Model," Mar 2024.[Online]. Available: <https://www.labellerr.com/blog/food-recognition-and-classification-using-deep-learning/>
- [3] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, and Y. Ma, "DeepFood: Deep Learning-Based Food Image Recognition for Computer-Aided Dietary Assessment," The University of Massachusetts, 2016. [Online]. Available: <https://arxiv.org/abs/1606.05675>
- [4] S. Rehan Shah, S. Qadri, H. Bibi, S. Muhammad Waqas Shah, M. Imran Ashraf, and F. Marinello, "Comparing Inception V3, VGG 16, VGG 19, CNN, and ResNet 50: A Case Study on Early Detection of a Rice Disease," Department of Computer Science, Muhammad Nawaz Shareef University of Agriculture Multan, 2023. [Online]. Available: <https://www.mdpi.com/2073-4395/13/6/1633>
- [5] D. Gomes, "Classification of Food Objects Using Deep Convolutional Neural Network Using Transfer Learning," University of Ulster, Belfast, Northern Ireland, 2023. [Online]. Available: <https://www.mecs-press.org/ijeme/ijeme-v14-n2/IJEME-V14-N2-5.pdf>
- [6] U. Sree Jakka, T. Bhagavan Reddy, P. Amulya, Y. Shaik Yunus Ahmed, "Automated Food Image Classification Using Deep Learning," Dept of Electronics and Communications Engineering, Sree Vidyanikethan Engineering College, Tirupathi, 2022. [Online]. Available: <https://www.irjweb.com/AUTOMATED%20FOOD%20IMAGE%20CLASSIFICATION%20USING%20DEEP%20LEARNING%20APPROACH.pdf>
- [7] Y. Kawano, K. Yanai, "Food Image Recognition with Deep Convolutional Features," Department of Informatics, The University of Electro-Communications, Tokyo, 2014. [Online]. Available: https://ubicomp.org/ubicomp2014/proceedings/ubicomp_adjunct/workshops/CEA/p589-kawano.pdf
- [8] A. Antonov, "Food-11 image dataset", 2019. [Online]. Available: <https://www.kaggle.com/datasets/trolovich/food11-image-dataset/data>
- [9] A. Arora, "Understanding ResNets: A Deep Dive into Residual Networks with PyTorch" Oct 2023. [Online]. Available: <https://wandb.ai/amanarora/Written-Reports/reports/Understanding-ResNets-A-Deep-Dive-into-Residual-Networks-with-PyTorch--Vmlldzo1MDAxMTk5#the-problem-with-deep-networks>
- [10] P. Ruiz, "Understanding and visualizing DenseNets," Oct 2018. [Online]. Available: <https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a>
- [11] "VGG-16 | CNN model," Mar 2024. [Online]. Available: <https://www.geeksforgeeks.org/vgg-16-cnn-model/>