Analysis of Deep Learning Techniques for Object Recognition in Images

Pedro Azevedo (102567), Pedro Mendes (103028)
Department of Electronics, Telecommunications and Informatics, University of Aveiro
Course: Complements of Machine Learning, Course Instructor: Petia Georgieva Georgieva
Email: {pgca, pmiguelsilvamendes}@ua.pt
Work Load: 50% 50%

***Abstract*—This project presents the implementation and evaluation of deep convolutional neural networks on the CIFAR-10 image classification task. Three deep learning architectures: VGG16, ResNet34, and DenseNet were implemented and evaluated to compare their performance in image recognition tasks. The dataset was divided into training, validation, and test sets to monitor learning progress and generalization, with standard data augmentation and optimization techniques applied during training.**

Performance was calculated using accuracy, confusion matrices, ROC curves, among other metrics, to gain detailed insights into class prediction behavior. The results demonstrate how effectively each model classifies images and provide a comparative analysis of their strengths. This evaluation helps identify which architecture may be better suited for real-world computer vision scenarios.

***Index Terms*—CIFAR-10, Deep Learning, Image Classification, VGG16, ResNet34, DenseNet**

I. Introduction

Convolutional Neural Networks (CNNs) have become one of the most powerful tools in the field of computer vision. They're especially useful for image classification tasks, where the goal is to teach a computer to recognize and categorize images. In this project, we focused on using three popular deep learning models: VGG16, ResNet34, and DenseNet, on the CIFAR-10 dataset, which consists of small, labeled images across 10 different classess.

The main idea was to train these models using slightly different training setups so we could compare their performance. Along the way, we also looked at how well each model generalized to unseen data by using validation and test sets.

II. Motivation

There are many deep learning models, each one has its own design and purpose. VGG16 is known for its simplicity and deep layers, ResNet34 uses shortcut connections to make training easier, and DenseNet connects each layer to every other layer to improve efficiency.

Our motivation was to better understand which model might work best in different situations. Since image classification plays a big role in things like facial recognition, autonomous vehicles or medical diagnosis, so being able to choose the right model can make a big difference.

III. Problem Complexity

At first the CIFAR-10 dataset might seem simple, as it only has 10 classes and the images are small. But in reality, it can be a bit challenging. The images are low resolution, and some classes look very similar (like cats and dogs), making it tricky for models to get high accuracy.

On top of that, training deep networks isn't always smooth. There are issues like overfitting, slow learning, or unstable training that can affect results. That's why architectures like ResNet and DenseNet were created—to solve some of these problems. In this project, we looked at how these models handle the complexity of CIFAR-10 and how their design choices impact performance.

IV. Dataset Description

The CIFAR-10 dataset (Canadian Institute For Advanced Research) [1] is a very well-known dataset, used in computer vision and machine learning, especially for training and testing image classification models. It contains 60 000 images, which are then split into 50 000 images for training and 10 000 for testing. As the name indicates, it contains 10 different classes of images, with the images evenly distributed for each class. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

In Figure 1, we can see the distribution of images that each class contains.

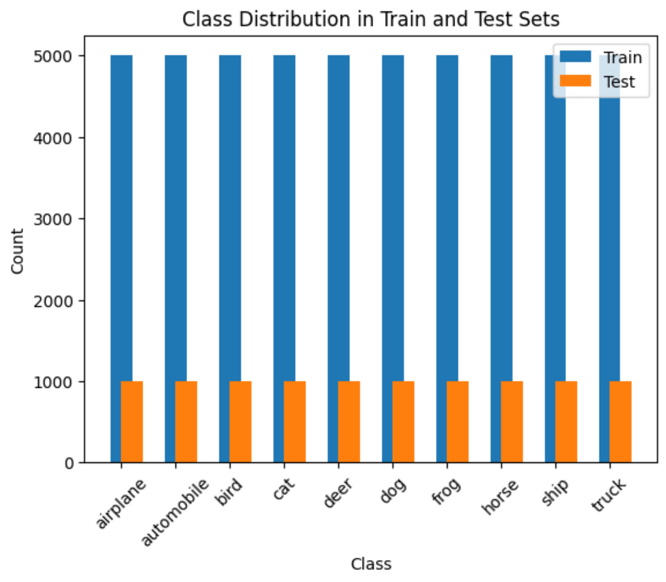


Fig. 1. Class distribution for the dataset's images

Each image in the dataset is a 32x32 pixels with 3 channels(RGB). To analyse the data itself, a boxplot was created to visualize the distribution of mean pixel intensities for each class. The mean pixel intensity is calculated as the average grayscale value of all pixels in an image. Figure 2 shows the distribution of these values across the classes.

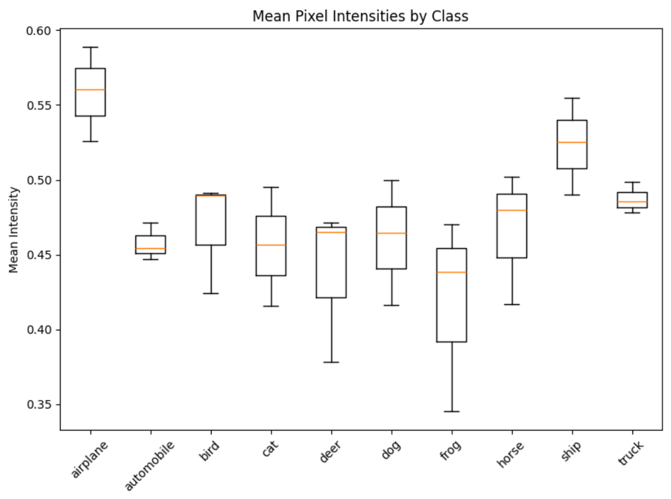


Fig. 2. Mean pixel intensity for each class

This plot provides valuable insights into the dataset. For instance, significant differences in the distributions between classes may make it easier for the machine learning algorithms to distinguish images and make more accurate predictions.

The dataset is split into three parts: Out of 60000 images, 50000 were used for training and validation in a 90/10 split, and the remaining 10000 were used for testing. This ensures that the models are evaluated on unseen data, which helps in assessing their generalization ability.

V. Methodology

As previously mentioned, three deep learning algorithms were used for this project, those being Visual Geometry Group Network (VGG16), Residual Network (ResNet34) and Dense Convolutional Network (DenseNet).

During training, we monitored the models' performance by checking the accuracy and loss for both the training and validation phases, while the final accuracy, loss, precision, recall and f1-score were calculated for the test set only. We also tracked the computation time required to run the entire algorithm.

A. VGG16

In this project, a VGG16 convolutional neural network was implemented and adapted for the CIFAR-10 dataset. The training data was augmented using random cropping and horizontal flipping to improve generalization, while both training and test sets were normalized using the dataset's mean and standard deviation.

The VGG16 model was loaded with pre-trained weights and modified to fit the CIFAR-10 dataset. Additionally, the training loop included a learning rate scheduler and optimization was performed using stochastic gradient descent with momentum and weight decay.

The following figures show the accuracy and loss values during the training and validation processes:

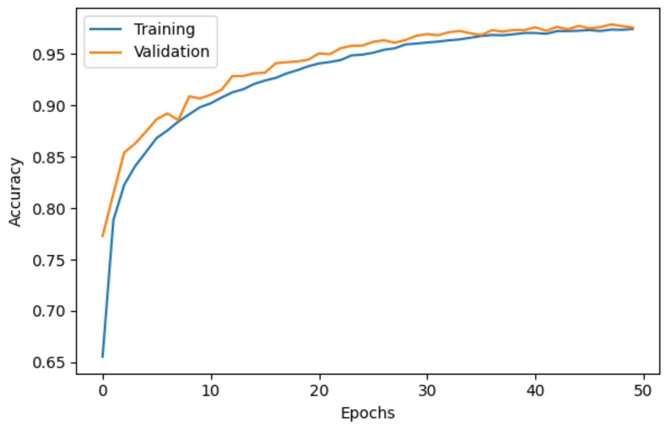


Fig. 3. Training and Validation Accuracy in the VGG16 model

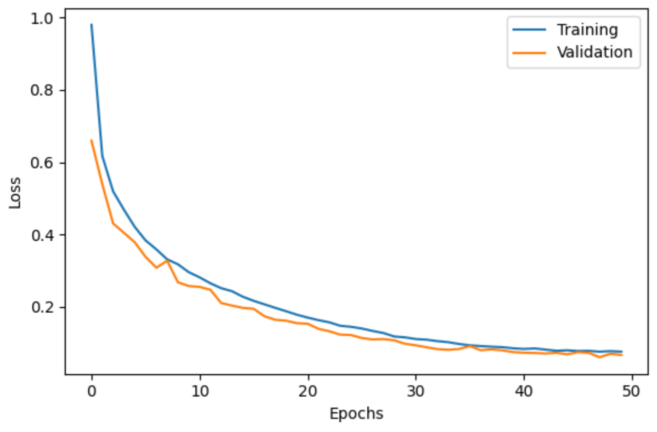


Fig. 4. Training and Validation Loss in the VGG16 model

The confusion matrix in Figure 5 summarizes the performance of the model by showing the true and predicted class distributions. For the most part, all classes have a reasonably good classification results, aside from classes *Cat* and *Dog*, which tend to be mistaken with one another a few times.

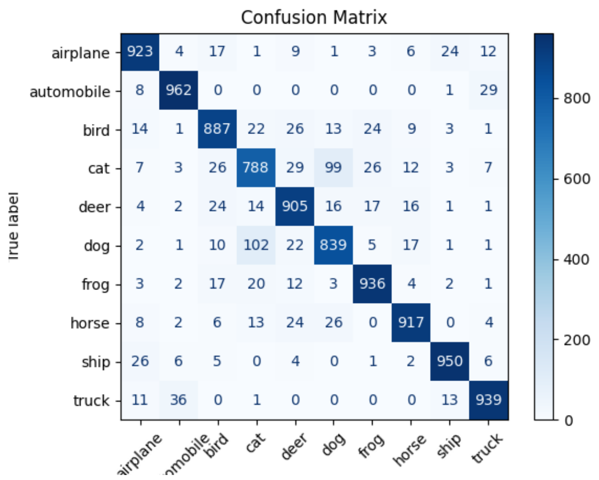


Fig. 5. Confusion matrix of the VGG16 model

The ROC curve in Figure 6 shows how well the model balances true positives against false positives for each class, where each class is compared with all the others. The Area Under the Curve (AUC) is used to measure the model's ability to distinguish between each class and all others, with values closer to 1 indicating almost perfect discrimination.

Grad-CAM is a visualization technique that highlights the regions in an input image that are most influential in a CNN's prediction. By using the gradients of the target class flowing into the final convolutional layer, Grad-CAM produces a coarse localization map, helping interpret and understand what parts of the image the model is focusing on during classification, as shown in Figure 17.

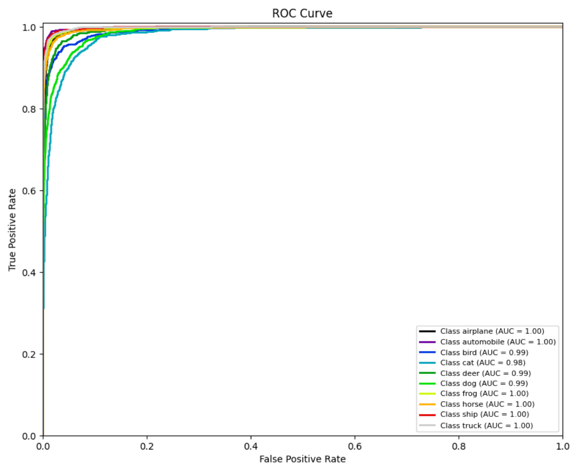


Fig. 6. ROC curve of the VGG16 model

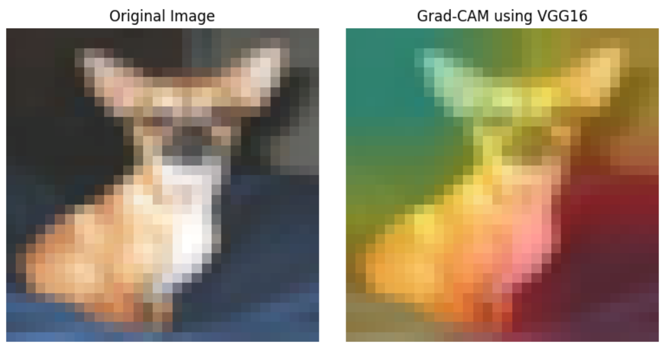


Fig. 7. Grad-CAM of the VGG16 model

B. *ResNet34*

Unlike the pre-trained VGG16 model, the ResNet34 model was trained from scratch. The model's architecture was adapted to fit the 32x32 images in the dataset. The first convolutional layer was modified to use a smaller kernel size and stride, and the initial max-pooling layer was removed. The final fully connected layer was replaced with a new linear layer with 10 output units, corresponding to the number of classes.

The model was trained using stochastic gradient descent (SGD) with momentum and weight decay and a cosine annealing scheduler was used to adjust the learning rate throughout the training process.

The following figures show the accuracy and loss values during the training and testing process, as well as the confusion matrix, ROC curve and the performance for each class.

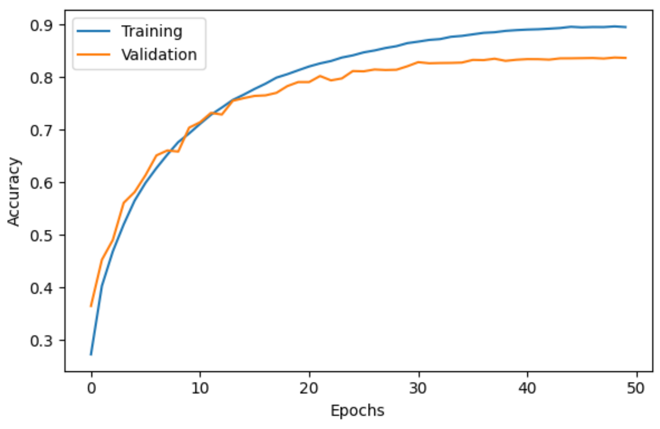


Fig. 8. Training and Testing Accuracy in the ResNet34 model

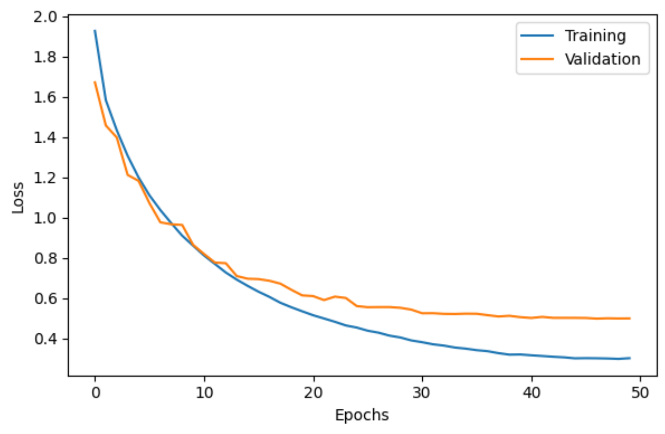


Fig. 9. Training and Testing Loss in the ResNet34 model

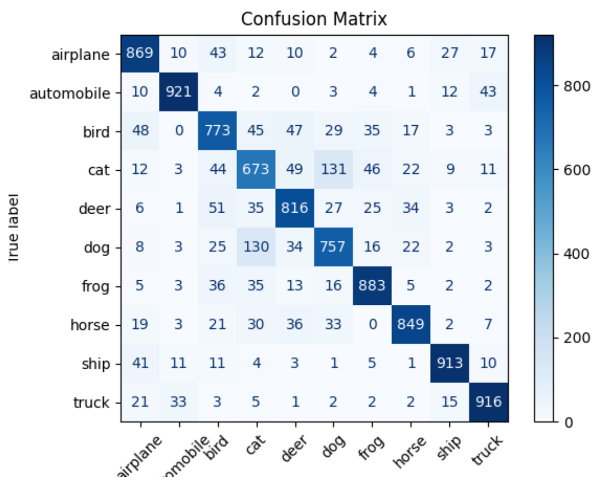


Fig. 10. Confusion matrix of the ResNet34 model

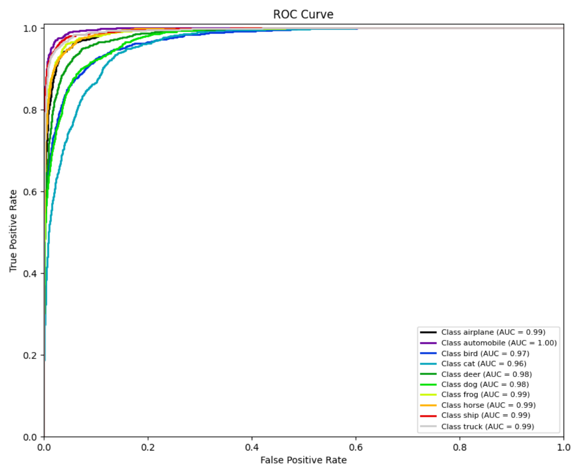


Fig. 11. ROC curve of the ResNet34 model

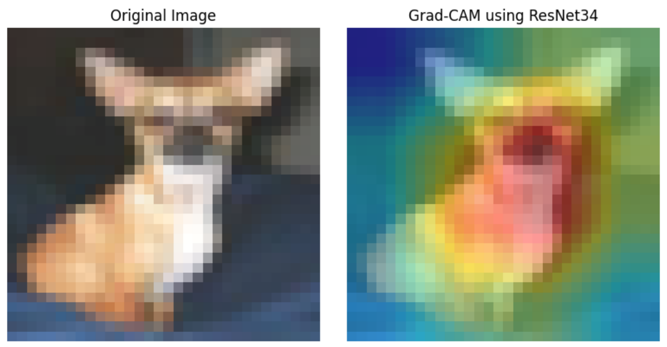


Fig. 12. Grad-CAM of the ResNet34 model

C. DenseNet

DenseNet (Dense Convolutional Network) is a type of deep learning architecture that is divided into blocks with several convolutional layers. Due to this long sequence of layers, the gradient of the errors (computed at the output) vanish before reaching the intial payers. There are multiple types of DenseNet, increasing in size from DenseNet-121 (which was the one used in our project up to DenseNet-264, which has more layers per block. This results in a higher parameter count, which leads to longer training time and higher memory use.

After a simple implementation of DenseNet121 that was fitted for 10 epochs, the model reached a test accuracy of 67%, while having over 90% of training accuracy. Since the images from CIFAR-10 are 32x32 pixels and do not match DenseNet default input shape, we built a model that would take this image size as default. Also, to avoid these signs of overfitting, data augmentation was implemented. This process included horizontal flips, width and height shifts, and small rotations up to 15 degrees. The model was then trained with the augmented data, and after some research it was it took a big number of epochs to train the model well. By looking through some notebooks, namely [2] and [3], where both models were trained for at least 50 epochs.

```
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)

datagen.fit(x_train)
```

After 120 epochs of training, these were the model's achieved metrics. The following figures show the accuracy and loss values during the training and testing process, as well as the confusion matrix, ROC curve and the performance for each class.

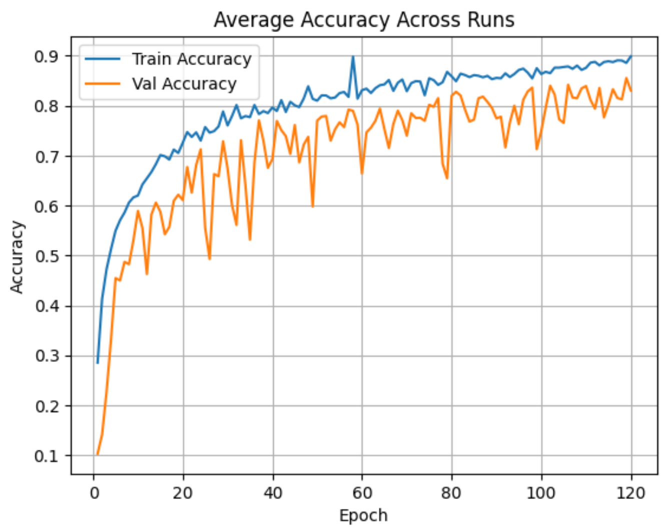


Fig. 13. Training and Testing Accuracy in the DenseNet model

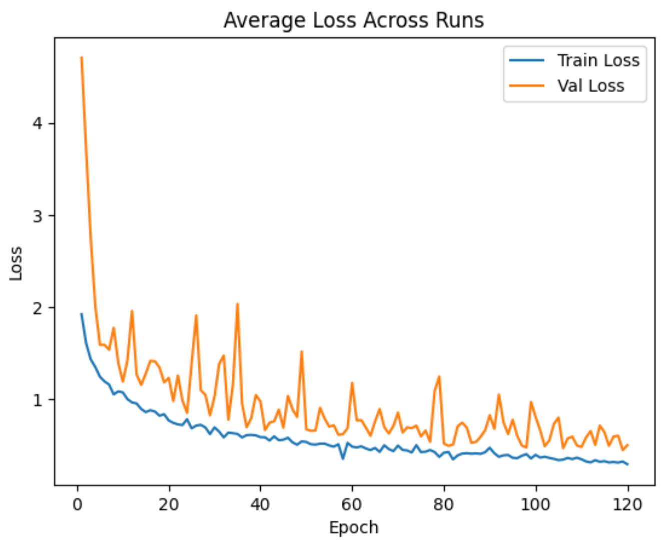
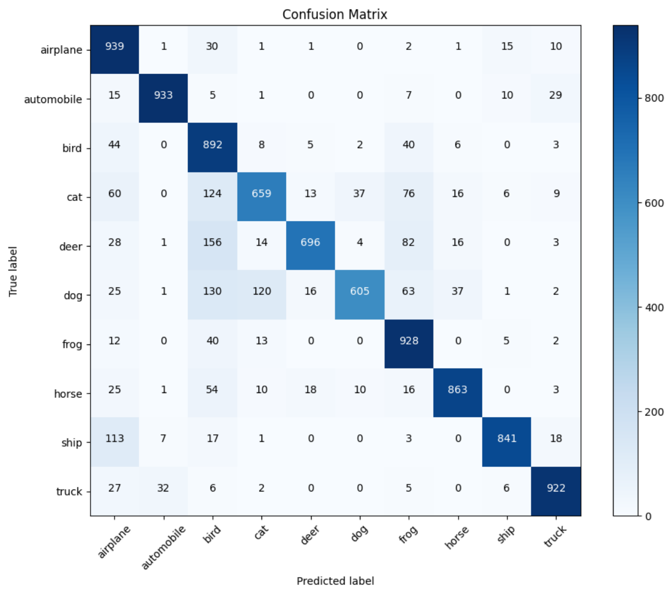


Fig. 14. Training and Testing Loss in the DenseNet model



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	939	1	30	1	1	0	2	1	15	10
automobile	15	933	5	1	0	0	7	0	10	29
bird	44	0	892	8	5	2	40	6	0	3
cat	60	0	124	659	13	37	76	16	6	9
deer	28	1	156	14	696	4	82	16	0	3
dog	25	1	130	120	16	605	63	37	1	2
frog	12	0	40	13	0	0	928	0	5	2
horse	25	1	54	10	18	10	16	863	0	3
ship	113	7	17	1	0	0	3	0	841	18
truck	27	32	6	2	0	0	5	0	6	922

Fig. 15. Confusion matrix of the DenseNet model

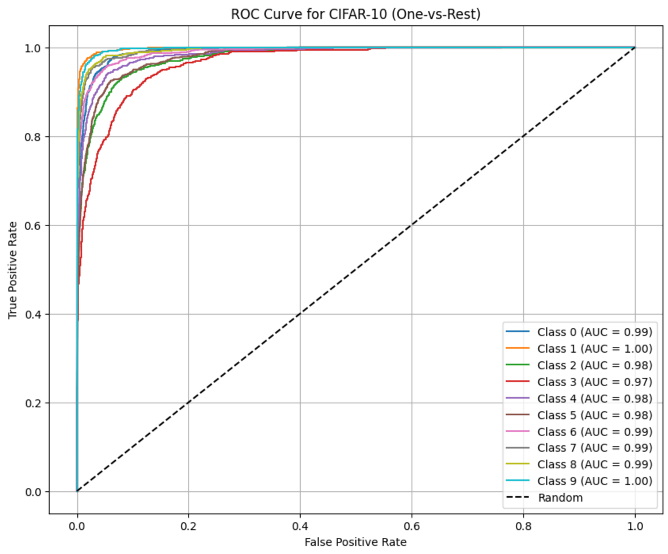


Fig. 16. ROC curve of the DenseNet model

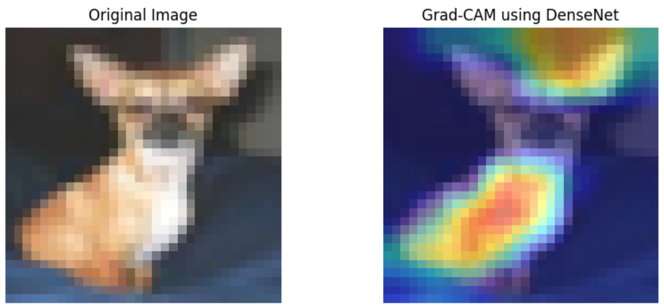


Fig. 17. Grad-CAM of the DenseNet model

VI. Results

The performance of each method was evaluated using accuracy, loss, precision, recall and f1-score. Table I presents the results obtained from all models.

TABLE I
Performance Metrics

Metric	VGG16	ResNet34	DenseNet
Accuracy	90.46%	83.26%	82.78%
Loss	0.3760	0.5203	0.5239
Precision	90.43%	83.19%	84.92%
Recall	90.46%	83.26%	82.78%
F1 Score	90.44%	83.21%	82.78%
Computation Time	3412.90s	4275.92s	5982.79s

The results demonstrate that all models performed reasonably well, with VGG16 outperforming both ResNet34 and DenseNet on all metrics. This performance is most likely due to being already pre-trained on ImageNet, which is a much bigger dataset than CIFAR-10.

As for ResNet34 and DenseNet, both performed somewhat similarly, with DenseNet having a slight underperformance compared to the other model.

VII. Comparison With Related Work

We've also compared our work with other author's work using the same dataset. In the Kaggle notebooks by Minh Khoa [4][5], the author used ResNet18 and ResNet50, in both cases using a different layer construction structure, with a few differences in the max pooling and dense layers, obtaining a better accuracy of 88.53% and 90.98%, respectively.

Another author is Mehmet Sahin [6], that used a VGG19 model that was also pre-trained on ImageNet, but for fewer epochs, resulting in an accuracy of 86.18%.

Both [2] and [3] implementations of DenseNet had better performance. [2] had more training, with around 200 epochs, and ended with around 90% accuracy.

[7] built a single DenseNet model for 2 different datasets, using CIFAR-10 for baseline performance the other dataset was the point of the study. His model performed with 68.9% accuracy on the CIFAR-10 dataset, a big underperformance comparing to the dataset on which the study was based that reached around 90% accuracy, although it has lower complexity, with just 3 classes.

[8] built a VGG16 model, that reached a 84% accuracy, which was an improvement compared to the CNN they built with 69% accuracy.

Finally, [9] built a VGG model, that took advantage of Feature Ensembles, and used PCA to select the more important features and reached an accuracy of 93.43%.

Further improvements to our models could involve a better layer building, properly adapted to our dataset. Changing the hyperparameters and data augmentation could also lead to a better performance.

VIII. Conclusion

This study compared VGG16, ResNet50, and DenseNet to solve an image classification problem. VGG16 showed a clear better performance, due to being already pre-trained. Nevertheless, both ResNet34 and DenseNet still had satisfying results, where the solutions mentioned previously would most likely lead to better results.

Deep learning has proven to be an invaluable tool in various fields, including image classification tasks like CIFAR-10. These models help automate complex tasks, reduce human error, and improve efficiency in interpreting large datasets, all while adapting to specific characteristics of the dataset for better performance.

References

[1] F. Nekouei, "Cifar-10 image classification with cnn," 2025, accessed: 28-04-2025. [Online]. Available: https://www.kaggle.com/code/farzadnekouei/cifar-10-image-classification-with-cnn
[2] "Densenet-on-cifar-10," 2018. [Online]. Available: https://github.com/aayushs879/Densenet-on-CIFAR-10/blob/master/DNST.ipynb
[3] "Densenet (tf-keras) on cifar-10," 2019. [Online]. Available: https://www.kaggle.com/code/asrsaiteja/densenet-tf-keras-on-cifar-10

[4] M. Khoa, “Resnet18 from scratch with adamw,” 2025, accessed: 02-05-2025. [Online]. Available: https://www.kaggle.com/code/vmkhoa28/resnet18-from-scratch-with-adamw

[5] ——, “Resnet50 from scratch with adamw,” 2025, accessed: 02-05-2025. [Online]. Available: https://www.kaggle.com/code/vmkhoa28/resnet50-from-scratch-with-adamw

[6] M. Sahin, “Cifar-10 /w vgg19,” 2025, accessed: 03-05-2025. [Online]. Available: https://www.kaggle.com/code/mehmet0sahinn/cifar-10-w-vgg19

[7] G. Wang, Z. Guo, X. Wan, and X. Zheng, “Study on image classification algorithm based on improved DenseNet,” *J. Phys. Conf. Ser.*, vol. 1952, no. 2, p. 022011, Jun. 2021.

[8] D. V. K. Samyal, H. Gupta, K. Karamveer, and D. Puri, “Transfer learning: Performance analysis of VGG-16 on CIFAR-10,” *Int. J. Res. Publ. Rev.*, vol. 5, no. 12, pp. 2742–2749, Dec. 2024.

[9] F. O. Giuste and C. Juan, *CIFAR-10 Image Classification Using Feature Ensembles*, 2020.