DEEP LEARNING PROJECT

|  |  |  |
| --- | --- | --- |
|  | ID | Full Name |
| 1 | 20211087 | يوسف عماد سيد على |
| 2 | 20210363 | زياد حسام حسن ابراهيم |
| 3 | 20211081 | يوسف عبدالمنعم حمدي عبدالمنعم |
| 4 | 20211054 | يسر حسن محمود ابراهيم |
| 5 | 20210371 | زياد محمد السيد نصر |
| 6 | 20210379 | زياد هاني احمد فرغلي |

**1-ResNet Architecture:**

**A residual neural network (also referred to as a residual network or ResNet) is a**[**deep learning**](https://uci.edog.cdn.office.net/wiki/Deep_learning)**architecture in which the layers learn residual functions with reference to the layer inputs. It was developed in 2015 for**[**image recognition**](https://uci.edog.cdn.office.net/wiki/Image_recognition)**, and won the**[**ImageNet**](https://uci.edog.cdn.office.net/wiki/ImageNet)**Large Scale Visual Recognition Challenge (ILSVRC) of that year.**

**Steps:**

**First step:**

**Data Preprocessing, prepare train & validation dataset using image\_dataset\_from\_directory.**

**Second step:**

**build resnet from scratch using three functions: 1-conv\_block,2-residual\_block,3-build\_resnet18.**

**Third step:**

**using checkpoint to save best weights using learning schedule.**

**fourth step:**

**training the model by 30 epochs.**

**Reference used:**

[**Resnet Architecture Explained. In their 2015 publication “Deep… | by Siddhesh Bangar | Medium**](https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d)

**Paper that introduced this architecture:**

**Paper=>** [**1512.03385**](https://arxiv.org/pdf/1512.03385)

**Graphs:**

**1-Accuracy&Loss visualization**

**A graph of a graph

Description automatically generated with medium confidence**

**2-Confusion matrix**

**A black and white graph

Description automatically generated with medium confidence**

**3-ROC Curve**

**A graph showing a curve

Description automatically generated**

**2-Xception finetuning Architecture:**

**Refers to the process of adapting a pre-trained Xception model to a new, specific task or dataset. This involves leveraging the model's existing learned features (trained on a large dataset like ImageNet) and modifying certain layers to make it suitable for the new problem.**

**Steps:**

**First step:**

**Data Preprocessing, prepare train & validation dataset using image\_dataset\_from\_directory.**

**Second step:**

**load xception model and make earlier layers trainable false and add some of layers in the end.**

**Third step:**

**using checkpoint to save best weights using learning schedule.**

**fourth step:**

**training the model by 30 epochs.**

**Reference used:**

[**Image Recognition using Pre-trained Xception Model in 5 steps | by Gopalakrishna Adusumilli | Analytics Vidhya | Medium**](https://medium.com/analytics-vidhya/image-recognition-using-pre-trained-xception-model-in-5-steps-96ac858f4206)

**Paper that introduced this architecture:**

**Paper=>** [**1608.06993v5**](https://arxiv.org/pdf/1608.06993v5)

**Graphs:**

**1-Accuracy&loss visualization**

**A comparison of a graph

Description automatically generated**

**2-confusion matrix**

**A graph of a diagram

Description automatically generated with medium confidence**

**3-ROC Curve visualization**

**A graph showing a curve

Description automatically generated with medium confidence**

**3-DenseNet finetuning Architecture:**

**Refers to the process of adapting a pre-trained DenseNet (Dense Convolutional Network) model for a specific task or dataset, using its existing feature extraction capabilities. DenseNet is characterized by its unique architecture where each layer is directly connected to every other subsequent layer, enabling efficient feature reuse and gradient flow.**

**Steps:**

**First step:**

**Data Preprocessing, prepare train & validation dataset using image\_dataset\_from\_directory.**

**Second step:**

**load DenseNet model and make earlier layers trainable false and add some of layers in the end.**

**Third step:**

**using checkpoint to save best weights using learning schedule.**

**fourth step:**

**training the model by 30 epochs.**

**Reference used:**

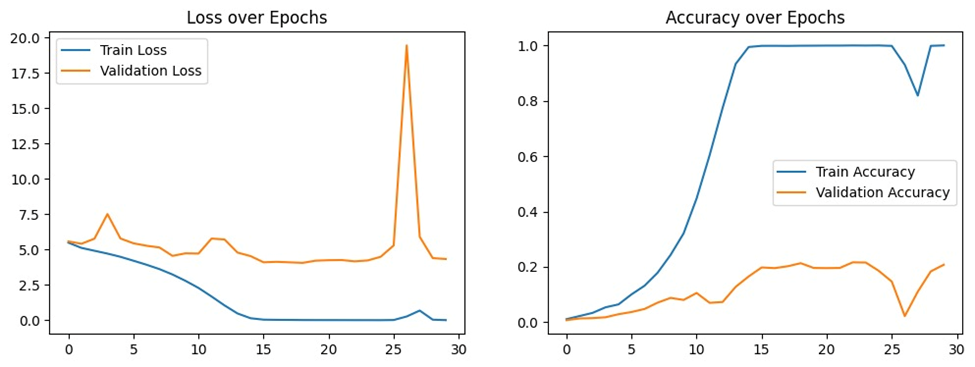
[**DenseNet : A Complete Guide. Extending the ResNet to improve… | by Alejandro Ito Aramendia | Medium**](https://medium.com/@alejandro.itoaramendia/densenet-a-complete-guide-84fedef21dcc)

**Paper that introduced this architecture:**

**Paper=>** [**1610.02357**](https://arxiv.org/pdf/1610.02357)

**Graphs:**

**1-Accuracy&loss visualization**

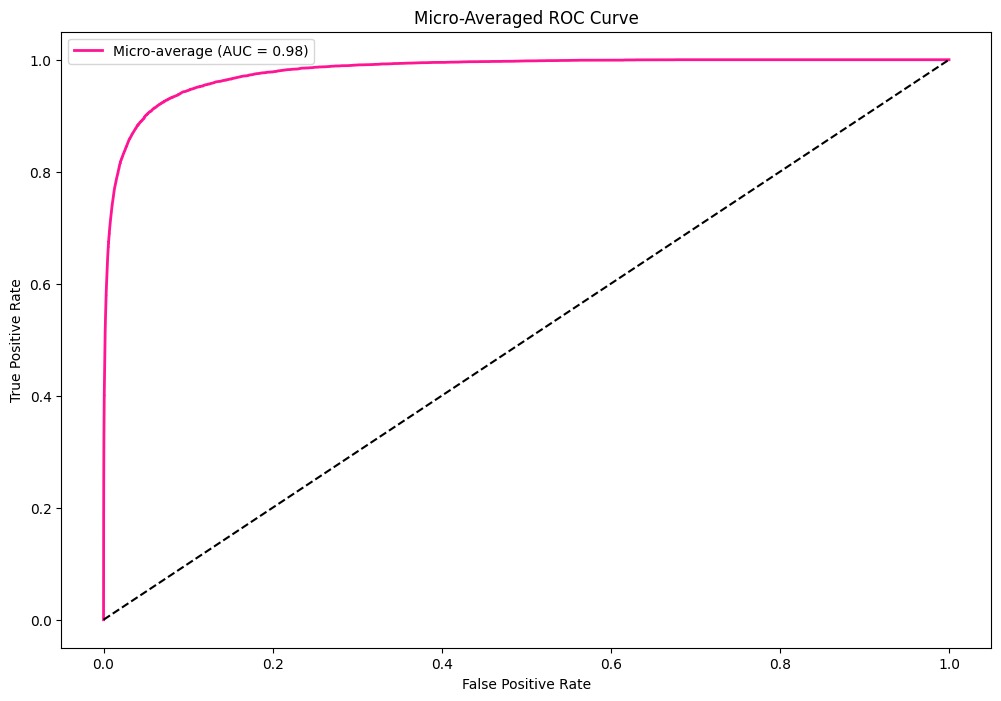
****

**2-confusion matrix**

**A diagram of a graph

Description automatically generated with medium confidence**

**3-ROC Curve**

****

Comparison between three architectures

|  |  |  |  |
| --- | --- | --- | --- |
| comparison | ResNet | Xception | DenseNet |
| result | Validation data: 20%  Test data: 19.75% | Validation data: 91%  Test data: 78% | Validation data: 51%  Test data: 53% |
| pros | 1-Simpler Architecture: Easier to debug and analyze compared to more complex models.  2-Smaller Size: Fewer parameters compared to Xception or DenseNet, leading to lower memory usage. | 1-Leverages pre-trained weights on ImageNet, leading to faster convergence and better performance with less data.  2-Good for Fine-Grained Data: The architecture excels in capturing subtle differences in car classes. | 1-Feature Reuse: Dense connections allow reusing features across layers, improving gradient flow and learning representation.  2-Good  Generalization: Handles overfitting well, even with relatively small datasets. |
| cons | 1-Data Hungry: Requires a large amount of data to achieve competitive performance.  2-Long Training Time: Training from scratch is computationally expensive and time-consuming.  3-Risk of Overfitting: More prone to overfitting on smaller datasets. | 1-Complex Architecture: More challenging to implement and debug due to depth wise separable convolutions.  2-Memory Intensive: Requires more computational resources compared to ResNet-18. | 1-High Computational Cost: Dense connections increase memory requirements during training.  2-Overhead in Layer Connections: The complex connection pattern can slow down forward and backward passes. |
| advantages | Best suited for scenarios where full control over model architecture and training is required.  Good for experiments to analyze the impact of architecture changes on performance. | Excels in scenarios with limited data availability but requires high accuracy.  Ideal for production-level applications due to their robust performance and relatively faster inference. | Great choice for applications where generalization is critical, especially with moderate computational resources.  Performs well in edge cases where similar-looking car classes need to be distinguished. |