Raw Materials Classification task using the TrashNet dataset

ResNet: helps with vanishing/exploding gradients problem that appear in training very deep neural networks.

Residual Block:

- A building block that uses shortcut connections to skip over layers.
- Each residual block consists of three convolutional layers with batch normalization and ReLU activation.
- If the input and output dimensions differ, a linear projection (via a 1x1 convolution) is applied to the shortcut connection.

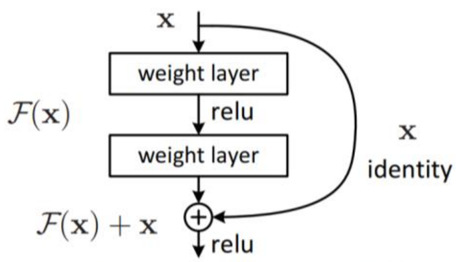


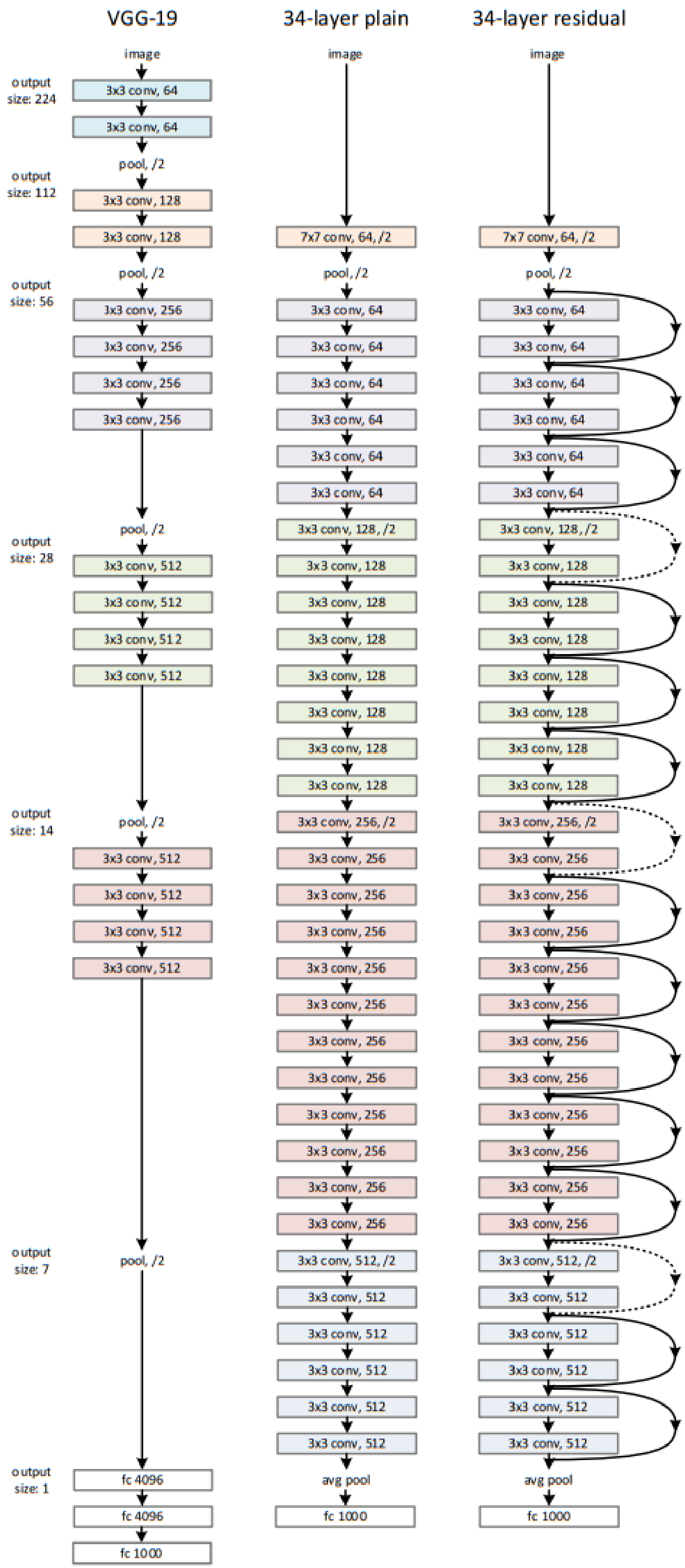
Figure 2. Residual learning: a building block.

ResNet-50: 50 layers (balanced performance)

The identity shortcut is used unless the block is downsampling.

<u>ResNet architecture:</u> Consists of three layers:

- 1x1 convolution -> dimensionality reduction.
- 3x3 convolution -> feature extraction.
- 1x1 convolution -> dimensionality restoration.



VGG-19
34-layer plain
34-layer residual
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1
image
3x3 conv, 64
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 128
pool, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
fc 4096
fc 4096
fc 1000
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

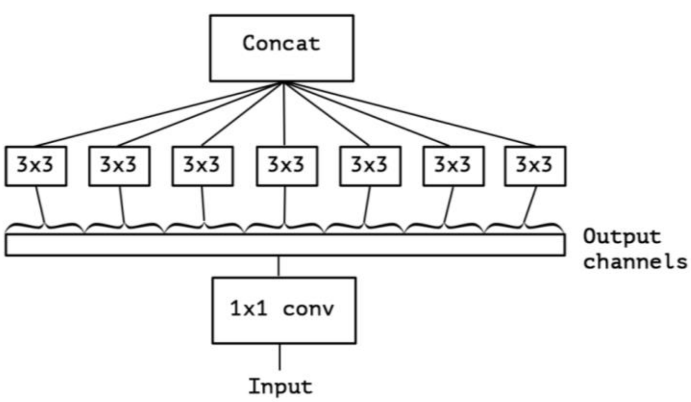
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Xception: extension of the Inception architecture, based on depthwise separable convolutions.

Depthwise Separable Convolution:

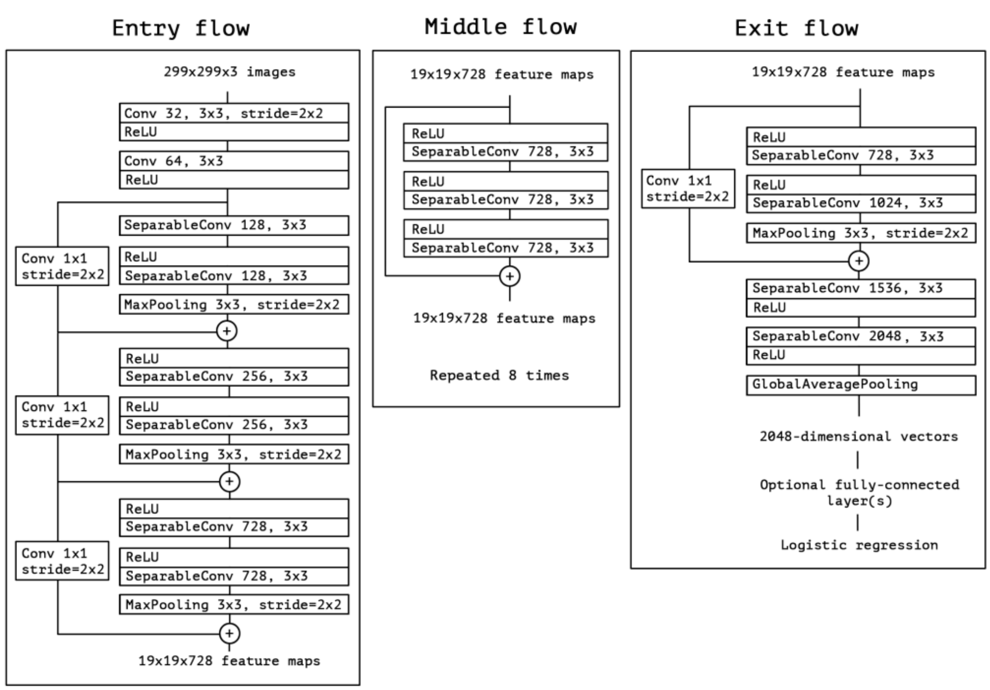
1. <u>Depthwise Convolution:</u> Apply a kernel to each input channel independently.
2. <u>Pointwise Convolution:</u> Combines the outputs from depthwise convolution using 1×1 convolutions.

Figure 4. An "extreme" version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.



<u>Xception Architecture:</u>

1. <u>Entry Flow:</u>
    - Series of convolutional layers with strides to reduce spatial dimensions.
    - Uses depthwise separable convolutions with residual connections.
2. <u>Middle Flow:</u> core part. Learning part. focus on learning detailed patterns.
    - Consists of 8 identical modules, each containing depthwise separable convolutions.
    - Channels remain constant throughout this stage but the filters focus on deepening the feature representation.
3. <u>Exit Flow:</u>
    - Increases feature maps while reducing spatial dimensions.
    - Final output is global average pooled and passed to the classifier.



Densenet: connects each layer to every other layer in a feed-forward manner within the block bring up extensive information flow throughout the network.

Densenet architecture:

Dense blocks: consists of multiple convolutional layers, typically followed by batch normalization and a non-linear activation function (e.g., ReLU)

Transition Layer: used to connect dense blocks to reduce the spatial dimensions and the number of feature maps. consists of:

- Batch Normalization
- 1x1 Convolution: Reduces the number of feature maps.
- Average Pooling: reduces the spatial dimensions.

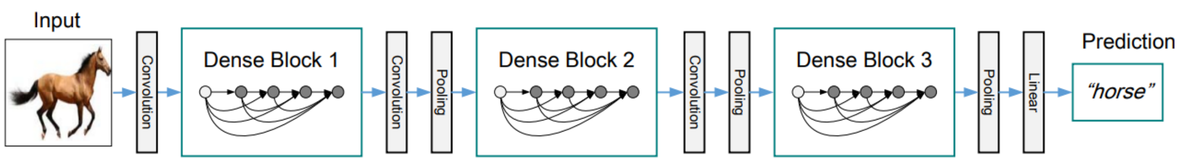


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

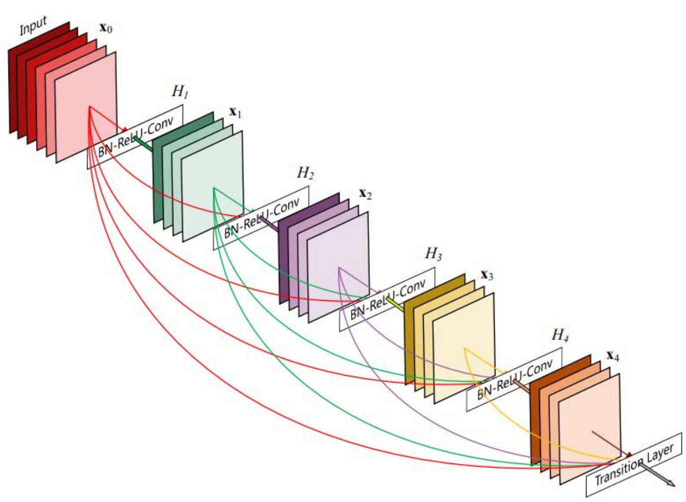


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Table 1: DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

Transhnet dataset: The dataset spans six classes: glass, paper, cardboard, plastic, metal, and trash. Currently, the dataset consists of 2527 images:

501 -> glass

594 -> paper

403 -> cardboard

482 -> plastic

410 -> metal

137 -> trash

The pictures were taken by placing the object on a white posterboard and using sunlight and/or room lighting. The pictures have been resized down to 512 x 384.

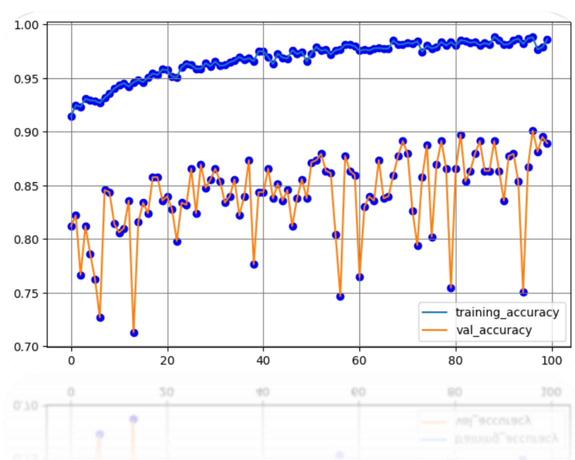
Summary:

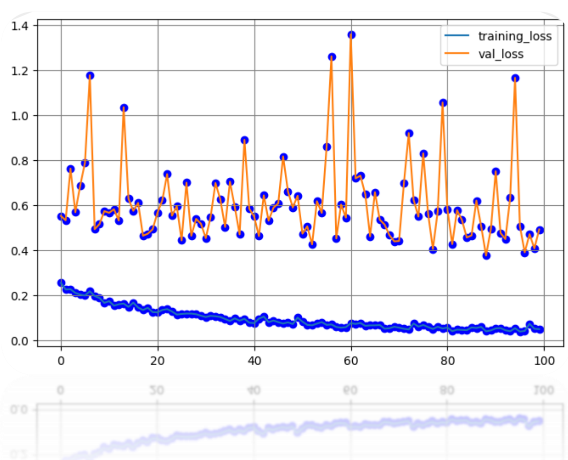
	ResNet	Xception	DenseNet
Results	Accuracy: 90.3% Precision: 0.9077 Recall: 0.9032 F1-Score: 0.9040 AUC: 0.9857	Accuracy: 93.7%	Accuracy: 97.1% Precision: 0.9689 Recall: 0.9684 F1-Score: 0.9684 AUC: 0.9971
Pros	- residual connections improve gradient flow and help with the vanishing gradient issues. - Simple to implement. - Faster convergence.	- Depthwise separable convolutions reduce parameters and computational cost. - High performance on large datasets. - More parameter-efficient than traditional models.	- Dense connections improve feature reuse and gradient flow. - Help with the vanishing gradient problem. - Parameter-efficient - Excellent performance with fewer parameters.
Cons	- Fine-tuning complex. - Not as parameter-efficient as DenseNet - high computational cost.	- Complex architecture and harder to implement. - Memory-intensive. - Fewer resources.	- High memory consumption led to slower training. - Complex architecture, harder to implement and debug.

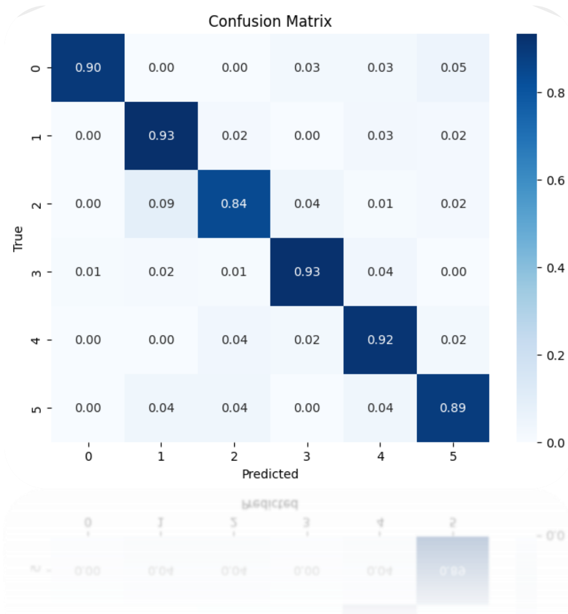
Advantages task-related	- Handles Complex Features in Raw Materials. - Improves Performance on Varied Data. - Reduces Risk of Overfitting.	- Efficient Texture Learning. - Reduces Computational Costs. - Handles Fine-Grained Classification.	- Captures Small Material Differences. - Efficient Use of Data. - Improved Accuracy with Fewer Parameters.

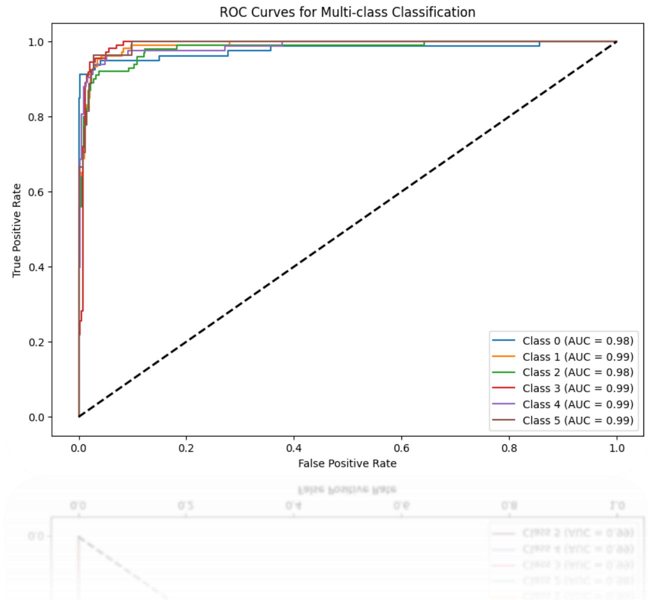
Graphs:

ResNet:

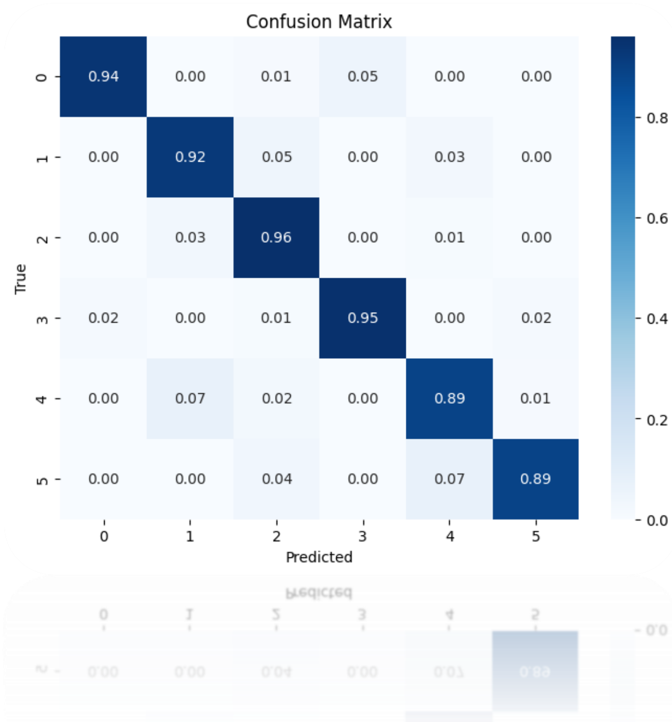




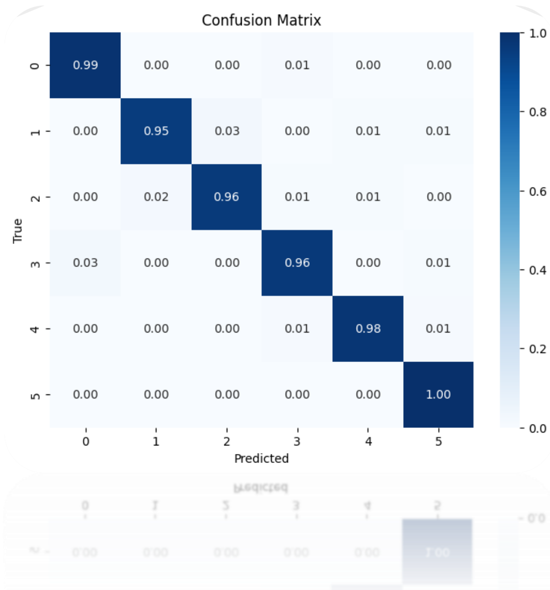


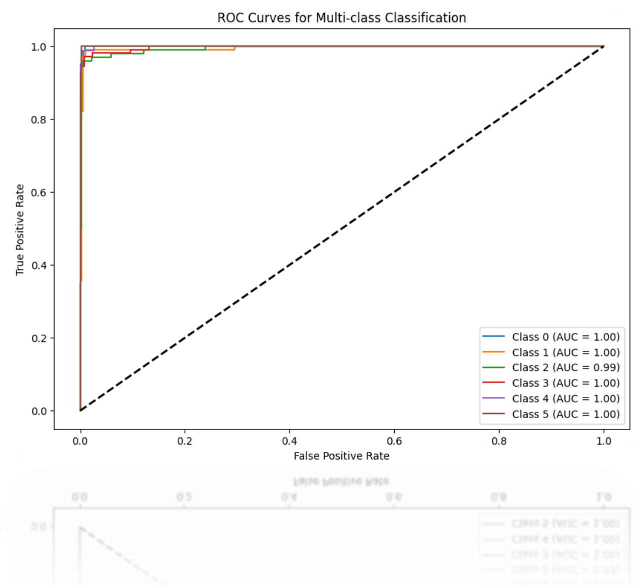


<u>Xception:</u>



<u>DenseNet:</u>


Confusion Matrix
True
Predicted
0.99 0.00 0.00 0.01 0.00 0.00
0.00 0.95 0.03 0.00 0.01 0.01
0.00 0.02 0.96 0.01 0.01 0.00
0.03 0.00 0.00 0.96 0.00 0.01
0.00 0.00 0.00 0.01 0.98 0.01
0.00 0.00 0.00 0.00 0.00 1.00


ROC Curves for Multi-class Classification
True Positive Rate
False Positive Rate
Class 0 (AUC = 1.00)
Class 1 (AUC = 1.00)
Class 2 (AUC = 0.99)
Class 3 (AUC = 1.00)
Class 4 (AUC = 1.00)
Class 5 (AUC = 1.00)

References:

pawangfg, P, Pawangfg, & Follow. (2023, January 10). Residual networks (resnet) - deep learning. GeeksforGeeks. https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

Boesch, G. (2024, December 10). XCEPTION model: Analyzing depthwise separable convolutions. viso.ai. https://viso.ai/deep-learning/xception-model/

GeeksforGeeks. (2024, June 6). DenseNet explained. https://www.geeksforgeeks.org/densenet-explained/

Papers:

ArXiv:1512.03385v1 [CS.CV] 10 dec 2015. Deep Residual Learning for Image Recognition. (n.d.). https://arxiv.org/pdf/1512.03385

Chollet, F. (2017, April 4). Xception: Deep learning with depthwise separable convolutions. arXiv.org. https://arxiv.org/abs/1610.02357

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2018, January 28). Densely connected Convolutional Networks. arXiv.org. https://arxiv.org/abs/1608.06993