1. ResNet (Residual Network)

Architecture

- Composed of **basic residual blocks**, where the output of a layer is added to its input.
- Each residual block consists of two convolutional layers, batch normalization, and ReLU activation.
- Structure:
    - Initial Conv layer.
    - 4 stages of residual blocks (2 blocks per stage with increasing filters).
    - Global Average Pooling (GAP) and Fully Connected (FC) layer.

Use Cases:

- Image classification tasks.
- Transfer learning on small to medium datasets.

group name	output size	block type = $B(3,3)$
conv1	32×32	$[3\times3, 16]$
conv2	32×32	$\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$
avg-pool	1×1	$[8 \times 8]$

Resnent Paper

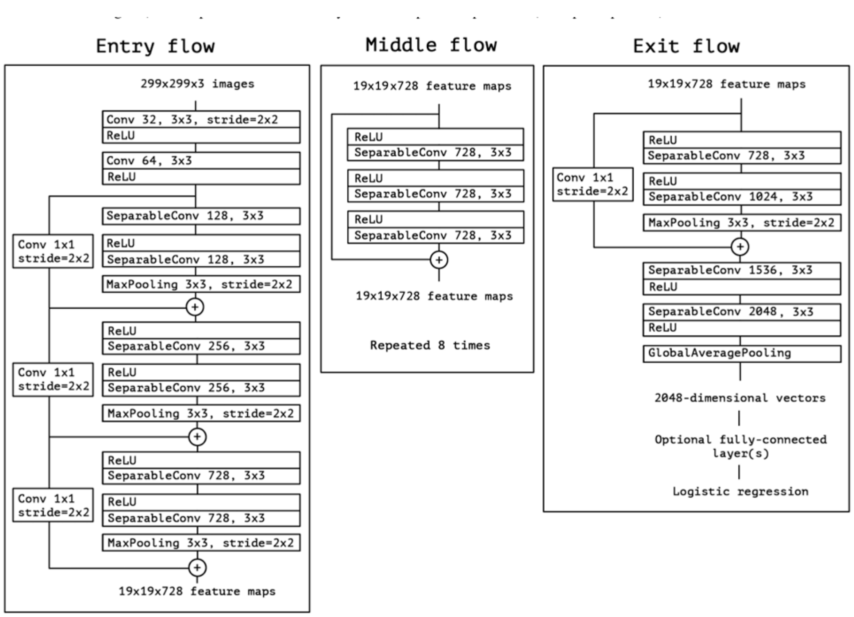
1. **Advantages**:
    a. **Solves vanishing gradient problem**: The residual connections allow gradients to flow directly through the network, enabling the training of very deep networks (e.g., ResNet-50, ResNet-101).

    b. **Ease of optimization**: Residual blocks simplify training by making it easier for the optimizer to fine-tune.
    c. **Wide adoption**: ResNet is a general-purpose architecture, effective for image recognition, object detection, and segmentation tasks.
    d. **Scalability**: Can be scaled to extreme depths (e.g., ResNet-152) without significant degradation in performance.
2. **Best for**:
    a. Tasks requiring deep networks with a focus on accuracy.
    b. Applications where computational resources are moderately constrained.

2. DenseNet-121 (Densely Connected Convolutional Network)

Architecture:

- Organized into **dense blocks**, each containing several convolutional layers.
- Between dense blocks, **transition layers** (with 1x1 convolutions and pooling) reduce feature map size and improve efficiency.
- Structure:
    - Initial Conv layer.
    - Dense blocks (with increasing growth rate of feature maps).
    - Transition layers between dense blocks.
    - GAP and FC layer.



Densnet Paper

Use Cases:

- Medical imaging tasks (e.g., cancer detection).
- Image classification for large datasets where feature reuse is beneficial.

3. Xception (Extreme Inception)

Architecture:

- Built entirely on **depthwise separable convolutions**:
  - Depthwise convolutions filter each input channel separately.
  - Pointwise convolutions combine filtered outputs.
- Includes residual connections for better gradient flow.
- Structure:
  - Entry flow: Initial feature extraction.
  - Middle flow: Deep feature learning with separable convolutions.
  - Exit flow: Classification layers.

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			

Xception Paper

Use Cases:

- High-performance image classification.
- Real-time object detection in resource-constrained environments.

- **Advantages**:
  - **Depthwise separable convolutions**: Replaces standard convolutions with depthwise separable convolutions, which reduce computational cost while improving performance.
  - **High model capacity**: Offers strong representation power due to its unique architecture.
  - **Optimized for large-scale tasks**: Performs particularly well on datasets with a high number of classes or complex patterns.
  - **Improved efficiency over Inception models**: Xception improves on the Inception family by achieving better performance with fewer parameters.
- **Best for**:
  - High-performance applications requiring efficient computation and high accuracy, such as mobile vision tasks or large-scale image classification.
  - Tasks where the computational complexity of DenseNet or ResNet may be prohibitive.

Dataset: Eminst letters

Description:
It contains handwritten English letters
(26 classes)

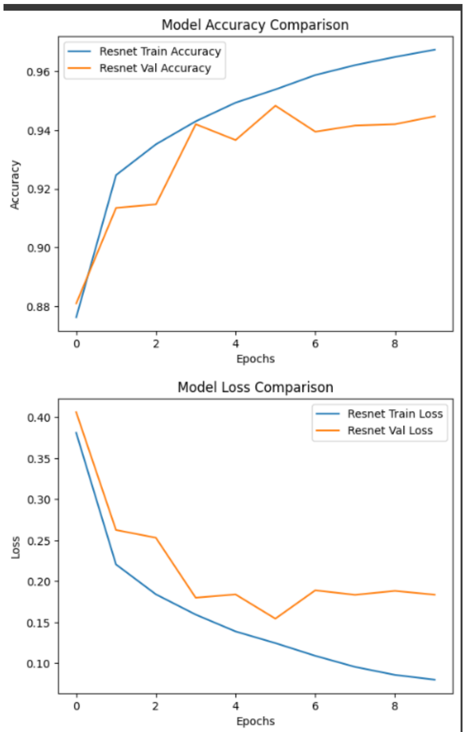
Shape: 28 , 28 , grey scale

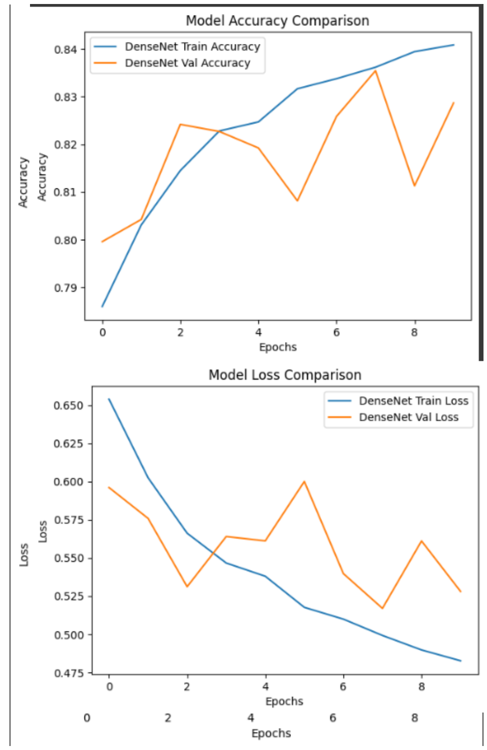
To be suitable for the models , should resizing it to 71 ,71 , RGB

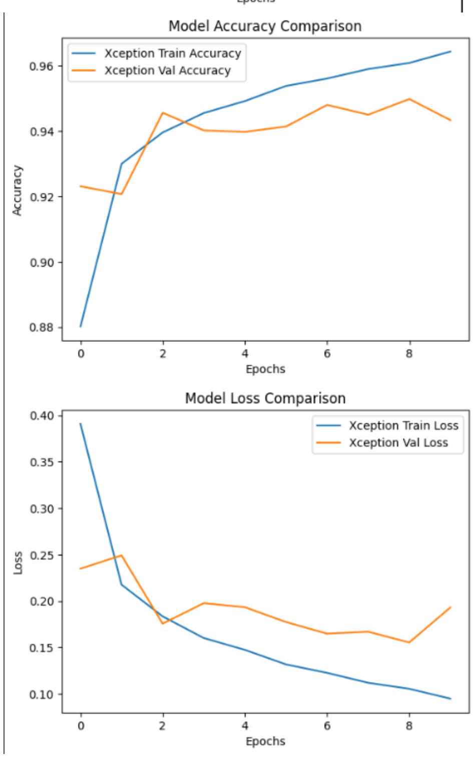
Comparison Of The Results Of The Three Models

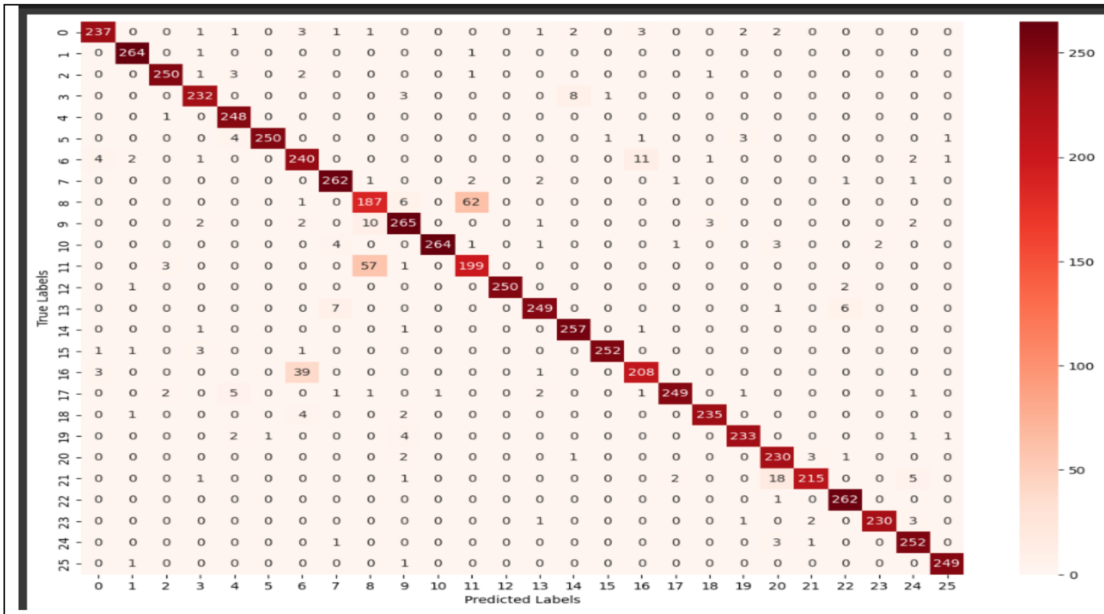
The loss and the accuracy of the three models

resnet densnet

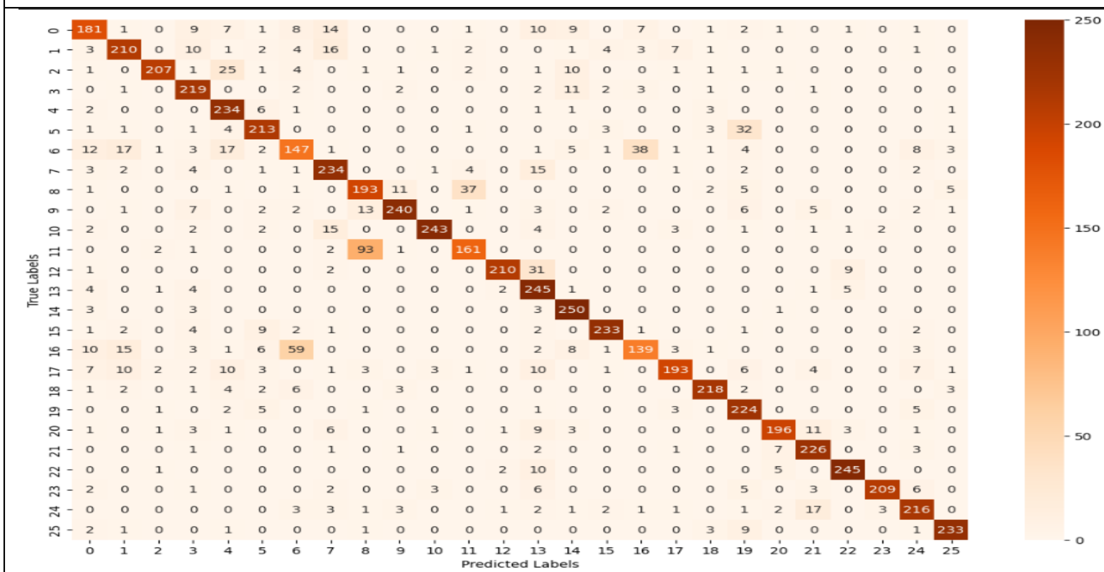

Model Accuracy Comparison
Resnet Train Accuracy
Resnet Val Accuracy
Accuracy
Epochs
Model Loss Comparison
Resnet Train Loss
Resnet Val Loss
Loss
Epochs


Model Accuracy Comparison
DenseNet Train Accuracy
DenseNet Val Accuracy
Accuracy
Epochs
Model Loss Comparison
DenseNet Train Loss
DenseNet Val Loss
Loss
Epochs

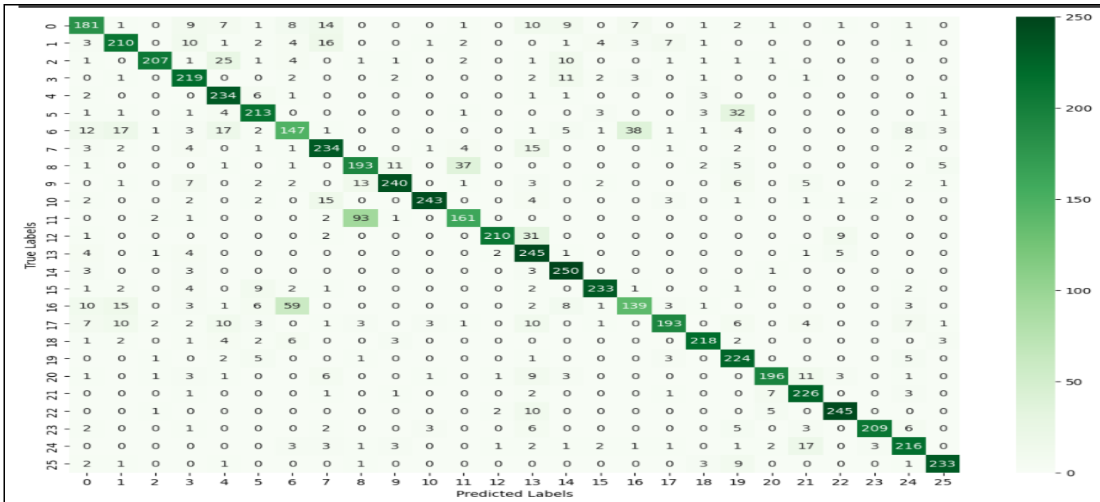

Model Accuracy Comparison
Xception Train Accuracy
Xception Val Accuracy
Accuracy
Epochs
Model Loss Comparison
Xception Train Loss
Xception Val Loss
Loss
Epochs



Resnet

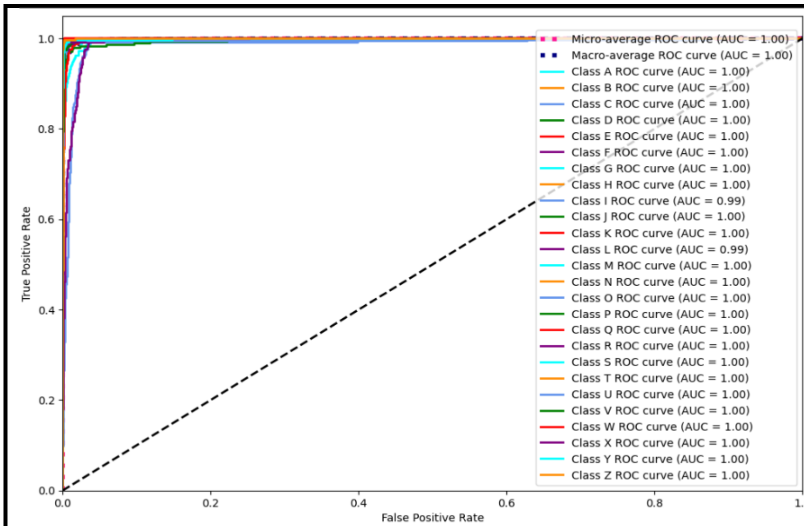


Densnet

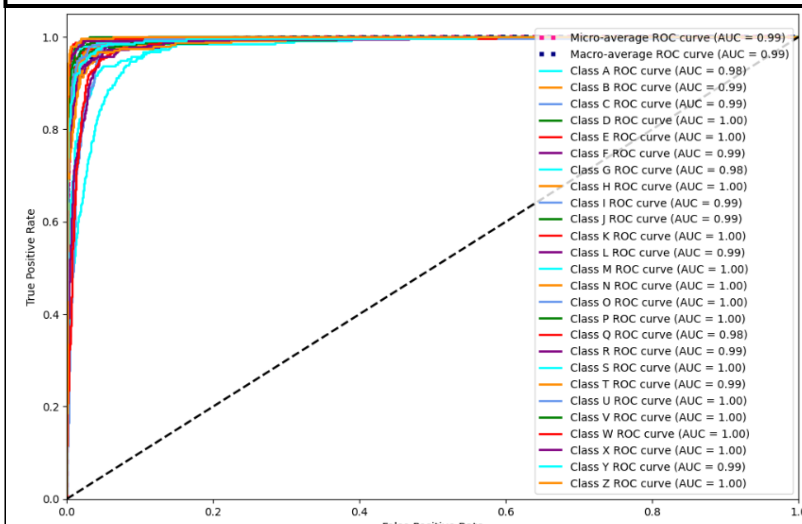


Xception

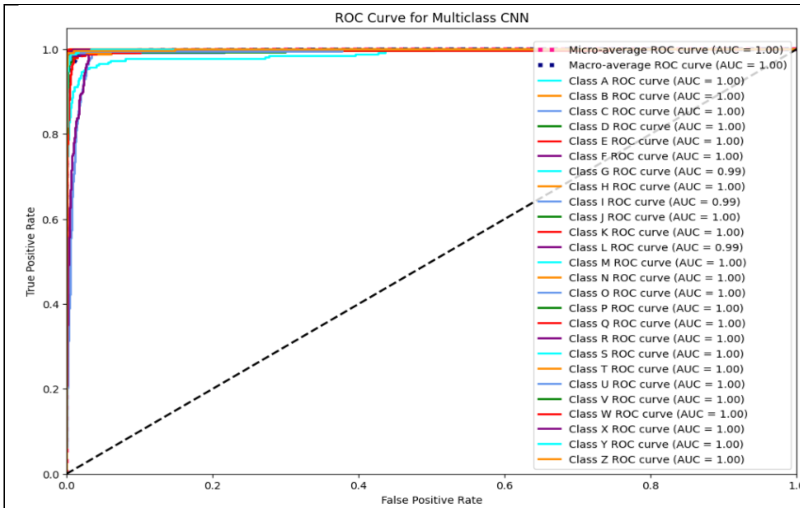
Roc Curve



Resnet



Densnet


ROC Curve for Multiclass CNN
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Micro-average ROC curve (AUC = 1.00)
Macro-average ROC curve (AUC = 1.00)
Class A ROC curve (AUC = 1.00)
Class B ROC curve (AUC = 1.00)
Class C ROC curve (AUC = 1.00)
Class D ROC curve (AUC = 1.00)
Class E ROC curve (AUC = 1.00)
Class F ROC curve (AUC = 1.00)
Class G ROC curve (AUC = 0.99)
Class H ROC curve (AUC = 1.00)
Class I ROC curve (AUC = 0.99)
Class J ROC curve (AUC = 1.00)
Class K ROC curve (AUC = 1.00)
Class L ROC curve (AUC = 0.99)
Class M ROC curve (AUC = 1.00)
Class N ROC curve (AUC = 1.00)
Class O ROC curve (AUC = 1.00)
Class P ROC curve (AUC = 1.00)
Class Q ROC curve (AUC = 1.00)
Class R ROC curve (AUC = 1.00)
Class S ROC curve (AUC = 1.00)
Class T ROC curve (AUC = 1.00)
Class U ROC curve (AUC = 1.00)
Class V ROC curve (AUC = 1.00)
Class W ROC curve (AUC = 1.00)
Class X ROC curve (AUC = 1.00)
Class Y ROC curve (AUC = 1.00)
Class Z ROC curve (AUC = 1.00)

Xception

	precision	recall	f1-score	support
0	0.97	0.93	0.95	254
1	0.98	0.99	0.99	266
2	0.98	0.97	0.97	258
3	0.95	0.95	0.95	244
4	0.94	1.00	0.97	249
5	1.00	0.96	0.98	260
6	0.82	0.92	0.87	262
7	0.95	0.97	0.96	270
8	0.73	0.73	0.73	256
9	0.93	0.93	0.93	285
10	1.00	0.96	0.98	276
11	0.75	0.77	0.76	260
12	1.00	0.99	0.99	253
13	0.97	0.95	0.96	263
14	0.96	0.99	0.97	260
15	0.99	0.98	0.98	258
16	0.92	0.83	0.87	251
17	0.98	0.94	0.96	264
18	0.98	0.97	0.98	242
19	0.97	0.96	0.97	242
20	0.89	0.97	0.93	237
21	0.97	0.89	0.93	242
22	0.96	1.00	0.98	263
23	0.99	0.97	0.98	237
24	0.94	0.98	0.96	257
25	0.99	0.99	0.99	251
accuracy			0.94	6660
macro avg	0.94	0.94	0.94	6660
weighted avg	0.94	0.94	0.94	6660

Resnet

	precision	recall	f1-score	support
0	0.76	0.71	0.74	254
1	0.80	0.79	0.79	266
2	0.96	0.80	0.87	258
3	0.78	0.90	0.84	244
4	0.76	0.94	0.84	249
5	0.84	0.82	0.83	260
6	0.61	0.56	0.59	262
7	0.79	0.87	0.82	270
8	0.63	0.75	0.69	256
9	0.92	0.84	0.88	285
10	0.96	0.88	0.92	276
11	0.77	0.62	0.69	260
12	0.97	0.83	0.90	253
13	0.68	0.93	0.79	263
14	0.83	0.96	0.89	260
15	0.94	0.90	0.92	258
16	0.72	0.55	0.63	251
17	0.90	0.73	0.81	264
18	0.93	0.90	0.91	242
19	0.74	0.93	0.83	242
20	0.92	0.83	0.87	237
21	0.84	0.93	0.88	242
22	0.93	0.93	0.93	263
23	0.98	0.88	0.93	237
24	0.84	0.84	0.84	257
25	0.94	0.93	0.93	251
accuracy			0.83	6660
macro avg	0.84	0.83	0.83	6660
weighted avg	0.84	0.83	0.83	6660

Densnet

	precision	recall	f1-score	support
0	0.97	0.91	0.94	254
1	0.91	1.00	0.95	266
2	0.99	0.96	0.97	258
3	0.93	0.96	0.95	244
4	0.96	0.99	0.98	249
5	0.98	0.97	0.98	260
6	0.97	0.71	0.82	262
7	0.97	0.98	0.97	270
8	0.69	0.84	0.76	256
9	0.94	0.95	0.95	285
10	1.00	0.99	0.99	276
11	0.83	0.67	0.74	260
12	1.00	1.00	1.00	253
13	0.96	0.98	0.97	263
14	0.98	0.98	0.98	260
15	0.98	0.98	0.98	258
16	0.81	0.97	0.88	251
17	0.98	0.97	0.97	264
18	0.99	0.98	0.99	242
19	0.97	0.99	0.98	242
20	0.92	0.97	0.94	237
21	0.96	0.92	0.94	242
22	0.98	0.99	0.98	263
23	0.98	0.98	0.98	237
24	0.98	0.95	0.96	257
25	0.98	0.99	0.99	251
accuracy			0.94	6660
macro avg	0.95	0.94	0.94	6660
weighted avg	0.95	0.94	0.94	6660

Xception

The accuracy of Each Model

Resnet	10 Epochs	94%
Densnet	10 Epochs	83%
Xception	10 Epochs	94%

Architecture

DenseNet

Pros

Highly efficient in parameters, improved gradient flow, compact representations

Cons

High computation and memory cost, potential overfitting on small datasets

ResNet

Pros
Stable training, widely supported, versatile, and effective for transfer learning

Cons

Computationally memory and intensive, diminishing returns with depth

Xception

Pros
Parameter-efficient, excellent performance,
strong generalization

Cons
Computationally expensive, less adoption for transfer learning, more complex to implement

Rating models on emnist dataset:

Resnet: it performs well with little overfitting

Densenet: it doesn’t perform well with overfitting

Xception: it performs well like Resnet with overfitting

models don’t have the best performance because resizing and changing images from grey scale to RGB.

So if we want the best performance we can use simple CNN model .

If we want to choose from these models priority for : Resnet because it is less complex than Xception
while Densent is excluded as it has the least performance .