➢ Overview of the models

Residual Networks (ResNet18)

- **Introduction:**

  Training deep neural networks often faces challenges such as vanishing gradients and a surprising degradation in accuracy as networks grow deeper. To address this, a framework was introduced that focuses on residual learning. Instead of learning a direct mapping from input to output, the model learns the residuals—essentially the differences—between the desired output and the input. This adjustment simplifies the learning process and makes it easier to train very deep networks, unlocking their full potential and achieving remarkable performance in image recognition tasks.

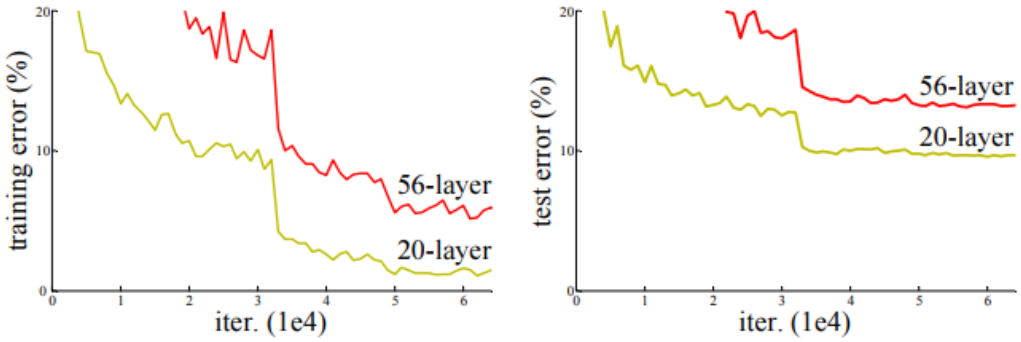


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- **Architecture:**
    1. **Core Concept**: Introduces shortcut connections, also known as skip connections, that bypass one or more layers in the network.

Residual Mapping: Instead of learning a direct mapping H(x)H(x)H(x), the network focuses on learning the residual F(x)F(x)F(x), where F(x)=H(x)−xF(x) = H(x) - xF(x)=H(x)−x. This formulation is restructured as H(x)=F(x)+xH(x) = F(x) + xH(x)=F(x)+x, allowing the network to easily refine its predictions.

- **Our Model's Achitecture:**

Stacked Blocks:

- Conv1: 3x3 convolution + Batch Normalization + ReLU + MaxPooling
- Conv2_x: 2 residual blocks with 64 filters.
- Conv3_x: 2 residual blocks with 128 filters.
- Conv4_x: 2 residual blocks with 256 filters.
- Conv5_x: 2 residual blocks with 512 filters.

Ends with global average pooling and a fully connected layer.

- **Advantages:**
    1. Alleviates the vanishing gradient problem, enabling more stable training of very deep networks.
    2. Facilitates the construction of significantly deeper networks without performance degradation.

- **Implementation:**
  1. **Optimization Method**: Utilizes Stochastic Gradient Descent (SGD) with momentum to improve convergence speed and stability during training.

  2. **Weight Initialization**: Adopts variance-scaled initialization to ensure proper scaling of weights, which helps prevent issues such as exploding or vanishing gradients.

  3. **Batch Normalization**: Applies batch normalization after every convolutional layer and before the ReLU activation function, standardizing inputs to each layer and accelerating training.

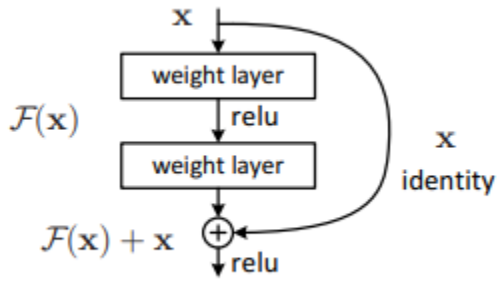
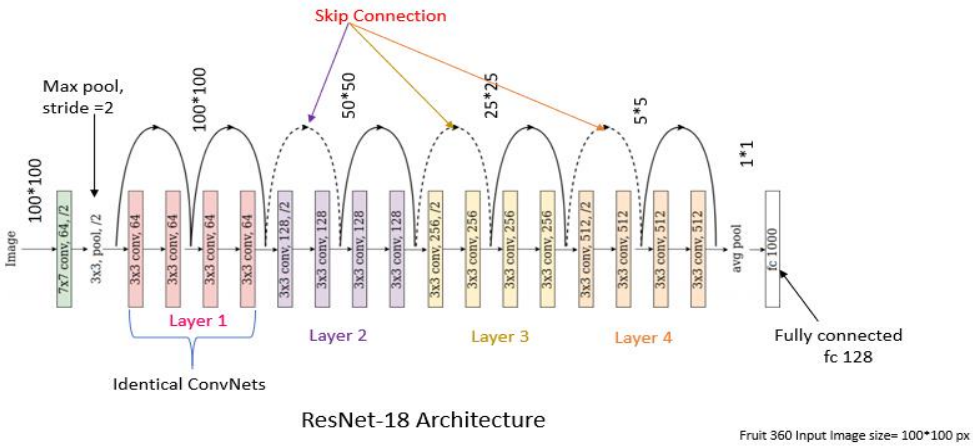


Figure 2. Residual learning: a building block.

- **Key Components:**

1. **Identity Shortcut:** Allows the gradient to flow directly to earlier layers.
2. **Residual Learning:** Encourages layers to learn the residual mapping instead of full mapping.



Pros:

- Efficient for tasks with limited computational resources.
- Easier optimization due to residual connections.
- Good performance on small datasets.

Cons:

- Underperforms on very large-scale tasks compared to deeper models.

DenseNet Model

DenseNet is a Convolutional Neural Network (CNN) designed to address challenges in deep learning models, such as:

1. Redundant feature extraction
2. Vanishing gradients in very deep networks.
3. Inefficient use of parameters.

DenseNet connects each layer to every other layer in a feed-forward fashion, ensuring maximum information flow.

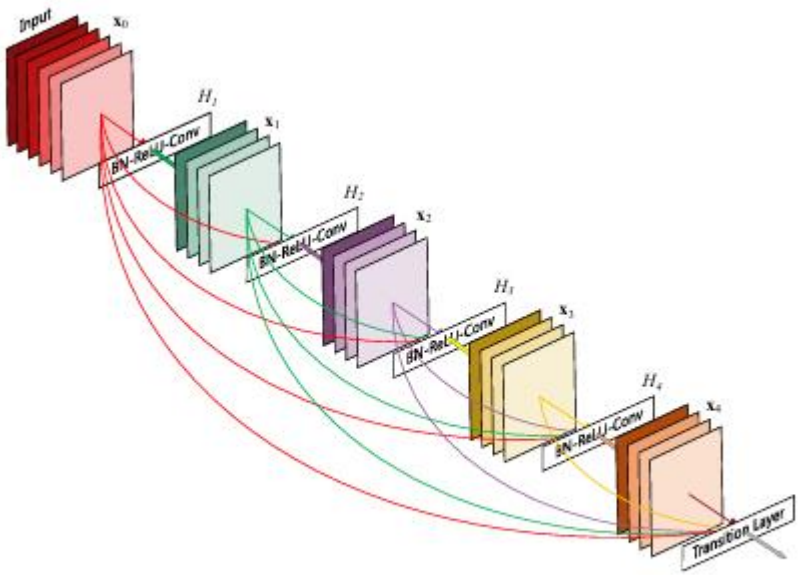


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Why DenseNet for EMNIST

The EMNIST dataset consists of simple grayscale images of handwritten letters. DenseNet's efficient feature reuse makes it particularly suitable for this dataset because:

1. The low complexity of EMNIST requires fewer parameters to achieve high performance.

2. DenseNet's feature concatenation promotes diverse representations for letters with similar shapes (e.g., "O" vs. "Q").

Step-by-Step DenseNet Architecture:

1. Input Layer :
   - For DenseNet121, the input size was originally 224×224 However, for EMNIST, the input size is resized to 64×64 to optimize for computational efficiency.
2. Dense Block :
   - **Batch Normalization (BN)**: Normalizes input to improve convergence.
   - **ReLU Activation**: Introduces non-linearity.
   - **3x3 Convolution**: Extracts spatial features.
   - The output of each layer is concatenated with all previous layers' outputs

3. Transition Layer :
   - ➢ Between dense blocks, a transition layer performs :
   - **1x1 Convolutions**: Reduces the number of feature maps.
   - **Pooling**: Downsamples the feature map dimensions.

4. Classification Layer **:**

After the final dense block, the model applies:

- **Global Average Pooling (GAP)**: Reduces the feature maps to a single vector per class.
- **Fully Connected Layer**: Outputs probabilities for the 26 EMNIST classes using a softmax function.

Implementation Details :

1. Data Preparation :
    - The input grayscale images (28×28) were resized to (64×64) and converted to RGB to match the pre-trained DenseNet121 architecture
    - Augmentation techniques like rotation, zooming, and flipping were applied for better generalization.
2. Model Configuration :

    The pre-trained **DenseNet121** was used with the following changes:

    - **Input Shape**: Adjusted to 64×64×3
    - **Output Layer**: Replaced with a dense layer with 26 units (corresponding to EMNIST classes).
    - **Dropout**: Added before the final layer for regularization

3. Optimization :
    - Used the **Adam optimizer** with a learning rate scheduler to adjust the learning rate during training.
    - Early stopping monitored validation loss to prevent overfitting.

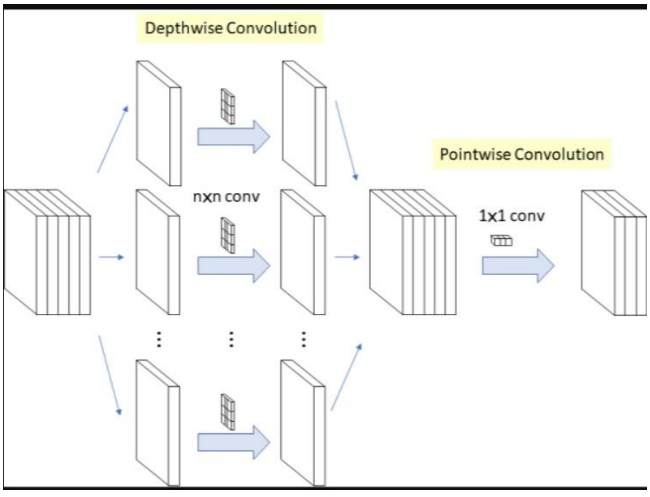
Extreme Inception (Xception)

- **Abstract:**

-- Xception model takes an even more aggressive approach as it entirely decouples the task of cross-channel and spatial correlation. This gave it the name Extreme Inception Model.

  - The Xception model, a pre-trained convolutional neural network (CNN), is considered a more effective and less complex solution to address the issues often encountered in various computer vision tasks. Many CNN architectures have been proposed for different applications, but they often suffer from high complexity, difficulty in extracting robust features, and degraded performance. Considering these problems with traditional CNNs, the authors present the Xception model as a promising alternative.
  - The Xception model's depth-wise separable convolutions and residual connections are designed to extract more effective features while maintaining a lower complexity compared to other CNN architectures. This makes the Xception model a compelling choice for a wide range of computer vision tasks, where the need for robust and efficient feature extraction is paramount.

Architecture:



The Xception model is characterized by several innovative architectural elements:

1. Depthwise Separable Convolutions: Xception utilizes depthwise separable convolutions, which separate the spatial and depth dimensions of the convolution operation, reducing parameters and computational cost.
2. Separable Convolution Blocks: Xception's architecture is based on separable convolutional blocks, each consisting of a depthwise separable convolution and a pointwise convolution. This modular design enhances the model's ability to capture complex patterns efficiently.
3. Hierarchical Structure: Xception is organized into an entry flow and an exit flow, where the entry flow extracts features, and the exit flow refines them for predictions. This hierarchical structure aids in learning hierarchical representations.
4. Skip Connections: Xception incorporates skip connections, enabling direct information flow across layers, which fosters efficient training of deep networks.

- ## Advantages:

-Xception has been shown to perform better than other architectures like VGG or ResNet on tasks such as image classification, particularly when fine-tuned on a smaller dataset like EMNIST.

- The depthwise separable convolutions reduce the number of parameters and the computation time compared to traditional convolutions.

- ## Implementation:

1)Load Xception base model:

- **The xception base model is loaded from keras library, without the top(classification) layer.**
- **The input shape is set to (71,71,3).**
- **The base model layers are frozen (meaning their weights are not updated during training).**

2) Build the model:

- **We created sequential model, then added global average pooling to reduce spatial dimensions.**
- **We added two dense layers with RelU activation and added two dropout to prevent overfitting.**
- **We added final dense layer with a softmax activation .**

3) train the model

- **We used early stopping and learning rate reduction to improve the training process.**
- **The model is trained using model.fit( ) .**

4) Use fine tuning:

- **Here we unfrozen the layer, starting from the last 20 layers.**
- **The model is recompiled with smaller learning rate for fine tuning.**

5) evaluate the model:

- **We evaluated the model on the test set.**

6) visualize the training and validation metrics:

- **We did it for fine tuning and the initial training.**

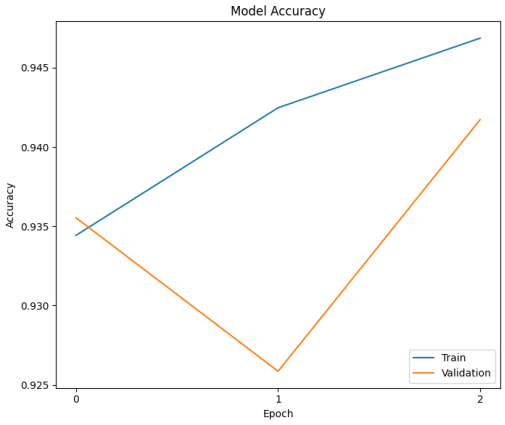
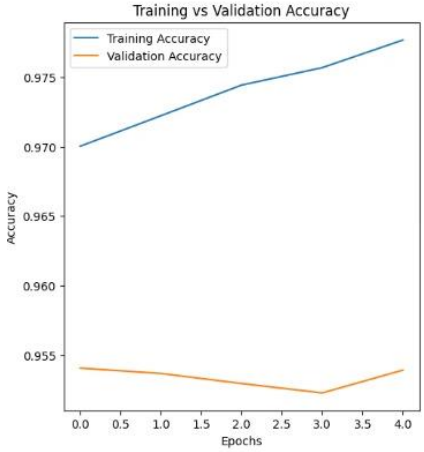
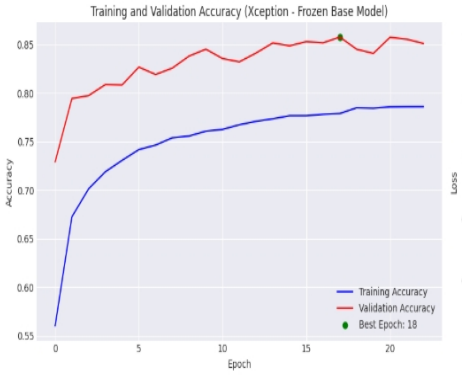
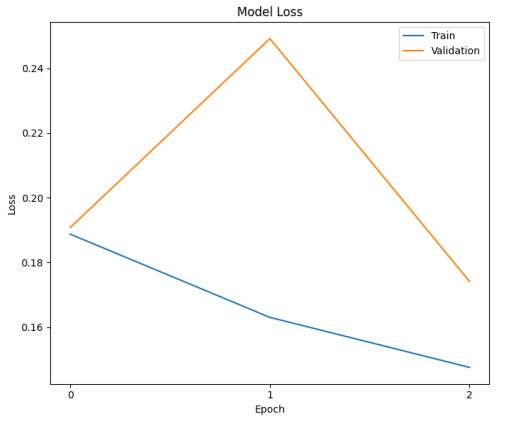
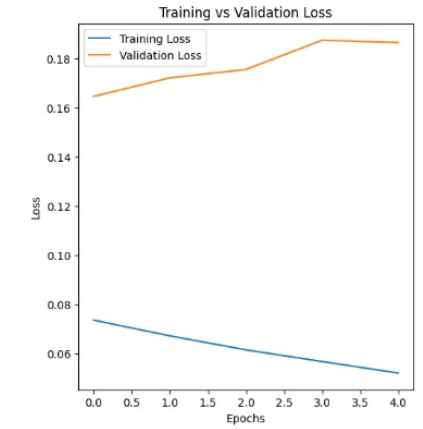
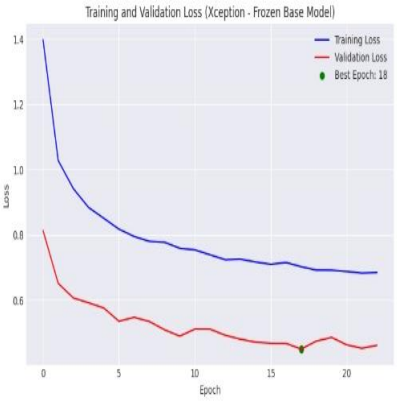
7): finally we plot the metrics as confusion matrix, precision, recall, F1 score, classification report providing detailed metrics for each class.

8) plot ROC curve:

- **The micro-average ROC curve and AUC (Area Under the Curve) are computed and plotted.**
- **The macro-average ROC curve and AUC are also computed and plotted for each class individually.**

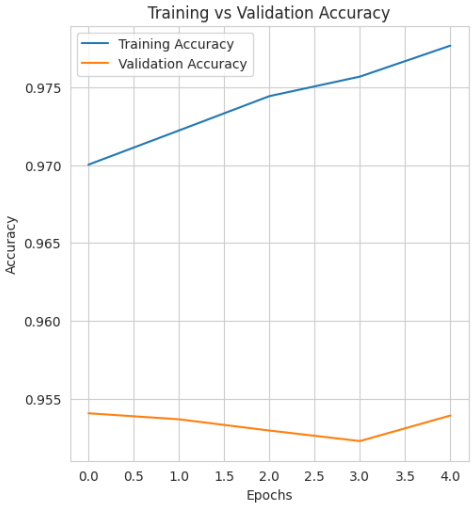
➢ Comparison among the models

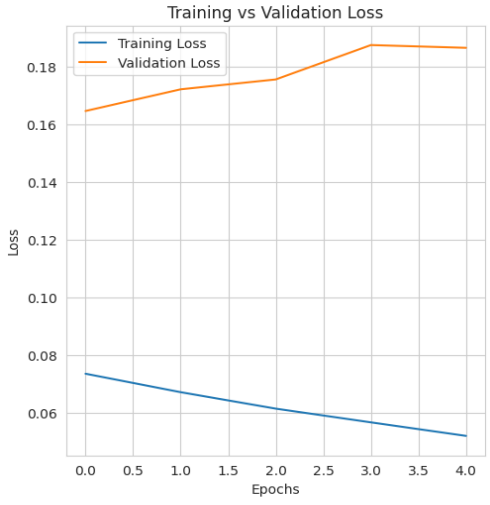
1. Accuracy & Loss curve

ResNet18	DenseNet	Xception
		
		

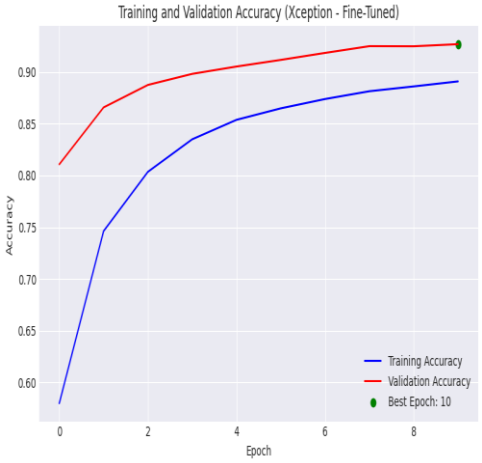
2. Accuracy & Loss curve

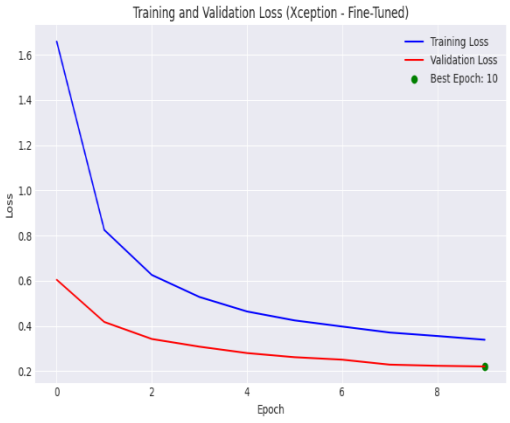
DenseNet





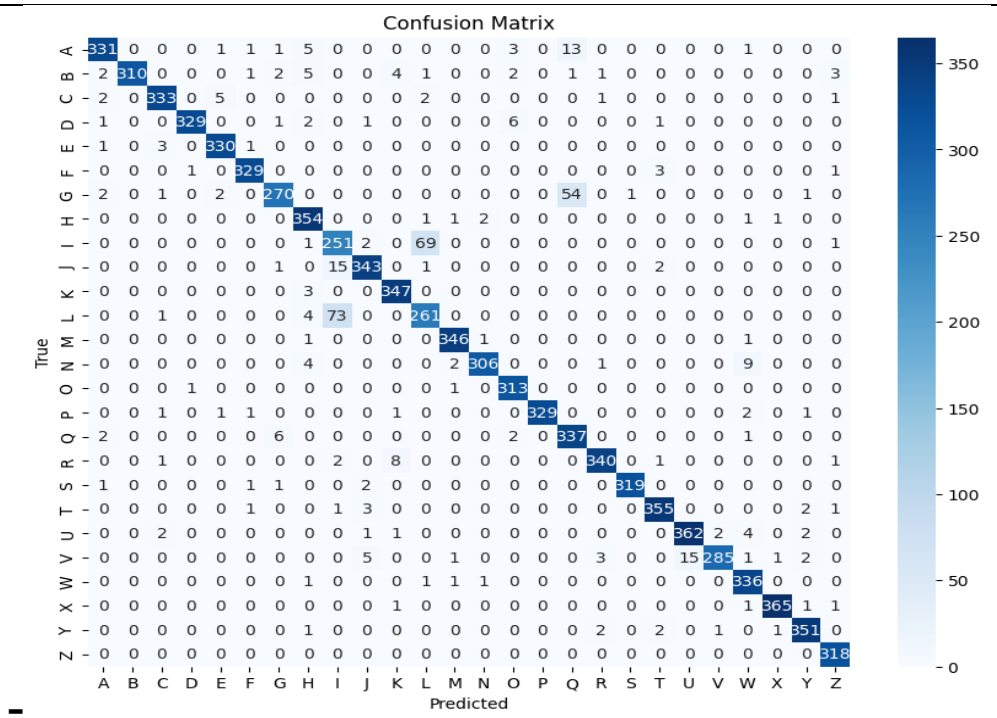
Xception



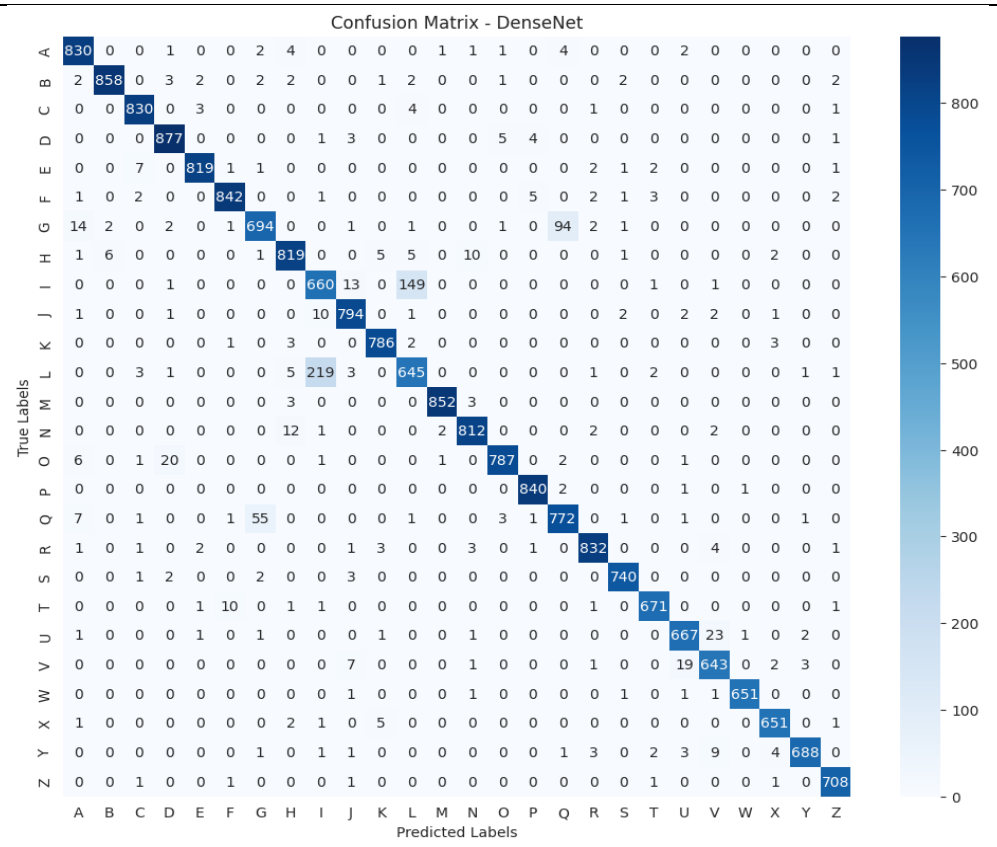


3. Confusion Matrix

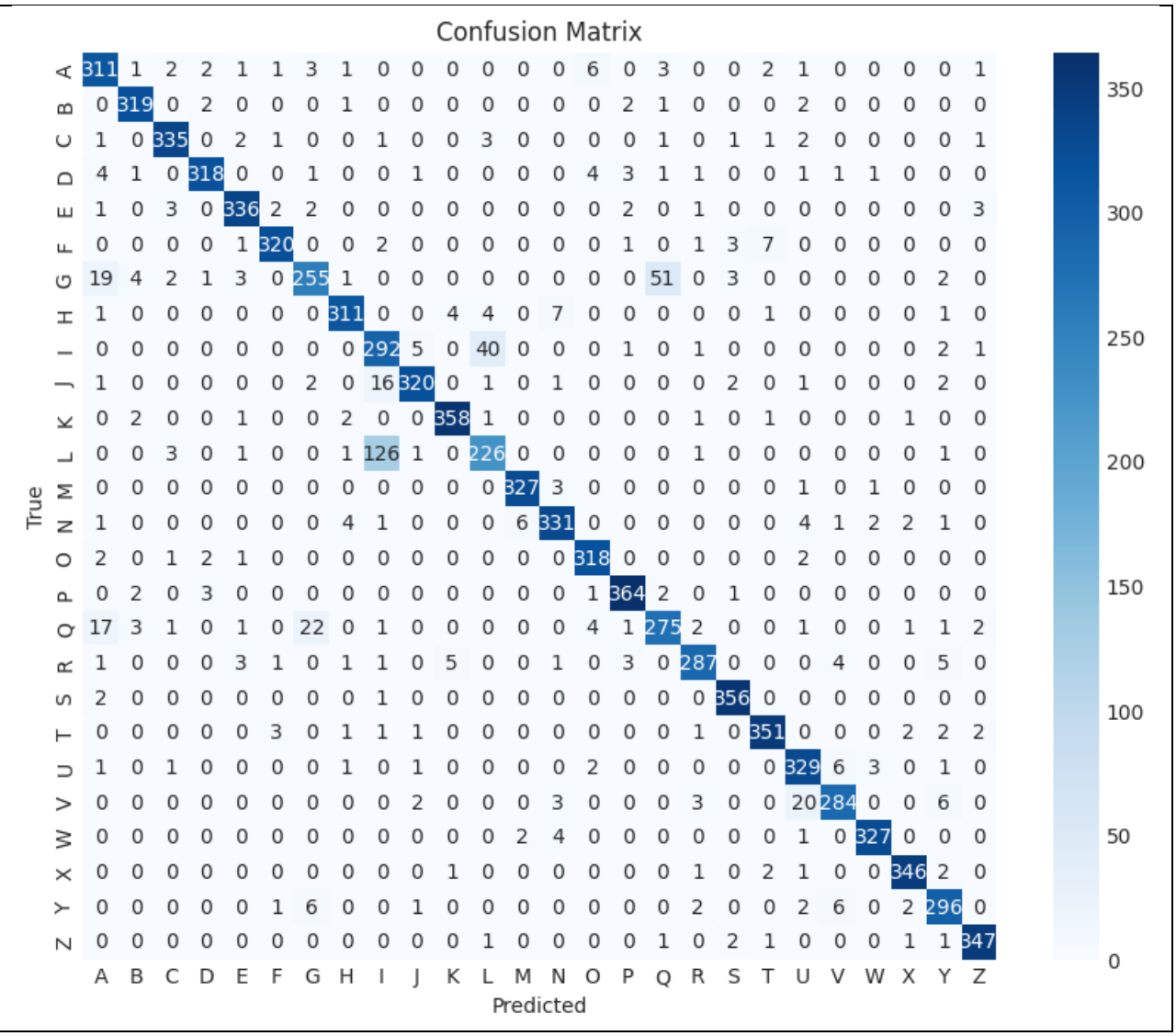
ResNet18:



DenseNet:



Xception:


Confusion Matrix
True
Predicted
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
350
300
250
200
150
100
50
0

Metric	Xception	ResNet-18	DenseNet-121	Xception Fine-tuning	DenseNet Fine-tuning
Accuracy	85.12%	94.17%	95.18%	92.6%	95.39%
Precision	___	95.21%	____	92.9%	95.43%
Recall	___	95.05%	____	92.7%	95.41%
F1-Score	___	95.06%	____	92.7%	95.4%

Architecture	Pros	Cons
ResNet-18	-Efficient for tasks with limited computational resources. -Easier optimization due to residual connections. -Good performance on small datasets.	-Underperforms on very large-scale tasks compared to deeper models.
Xception	-Efficient Use of Parameters: Xception replaces standard convolutions with depthwise separable convolutions, significantly reducing the number of parameters while maintaining or even improving performance. -Improved Accuracy: Its architecture is better at learning spatial correlations independently for each	Requires Powerful Hardware: Xception performs well on GPUs and TPUs

	channel, leading to better accuracy in various image classification and object detection tasks.	
DenseNet	**-Feature Reuse**: Efficiently reuses features across layers, reducing redundancy and improving performance. **-Parameter Efficiency:** Requires fewer parameters compared to similar architectures. -**Mitigates Vanishing Gradients:** Dense connections improve gradient flow, enabling deeper networks. **-Compact Model Size:** Offers strong performance with lower storage requirements. **-Good for Transfer Learning:** Feature-rich layers generalize well to other tasks.	- **High Memory Usage**: Dense connections increase memory demands, especially in deep networks. **-Computational Overhead**: Slower inference and training due to concatenation of feature maps. -**Scalability Challenges**: Becomes resource-intensive as network depth grows. **-Complex Implementation**: More challenging to implement than simpler architectures. **-Not Lightweight**: Less suitable for mobile or real-time applications.

➢ Observation:

Rating models on emnist dataset:

Resnet: it performs well with little overfitting

Xception: it doesn't perform well with overfitting

DenseNet: it performs well like Resnet with overfitting models.

if we want the best performance we can use simple DenseNet model . If we want to choose from these models priority for : DenseNet then ResNet, then Xception.

References:

Deep Residual Learning for Image Recognition

Xception: Deep Learning with Depthwise Separable Convolutions

Densely Connected Convolutional Networks