HUMAN TISSUE CLASSIFICATION

OVERVIEW

Classifying different types of human tissues, including distinguishing between normal and cancerous tissues (like neural or other types), is a crucial task in biomedical research and clinical diagnostics. This project aims to develop methods for accurately identifying tissue types based on NCT-CRC-HE dataset

DATASET BEST BENCHMARK

#SORTED BY ACCURACY

Model	accuracy	f1-score	precision	paper name
Efficientnet-b0	**95.59**	**97.48**	**99.89**	**EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks**
ResNeXt-50-32x4d	95.46	97.46	99.91	ResNet strikes back: An improved training procedure in timm
RegNetY-3.2GF	95.42	97.39	99.97	RegNet: Self-Regulated Network for Image Classification
ResNet-50	94.72	97.09	100.00	Deep Residual Learning for Image Recognition
DenseNet-169	94.41	96.90	99.87	Densely Connected Convolutional Networks

BEFORE DISCUSSING THE MODELS USED ON OUR DATASET, WE SHOULD DEFINE TRANSFER LEARNING

Transfer learning is a machine learning technique in which knowledge gained through one task or dataset is used to improve model performance on another related task and/or different dataset.1 In other words, transfer learning uses what has been learned in one setting to improve generalization in another setting.2 Transfer learning has many applications, from solving regression problems in data science to training deep learning models. Indeed, it is particularly appealing for the latter given the large amount of data needed to create deep neural networks.

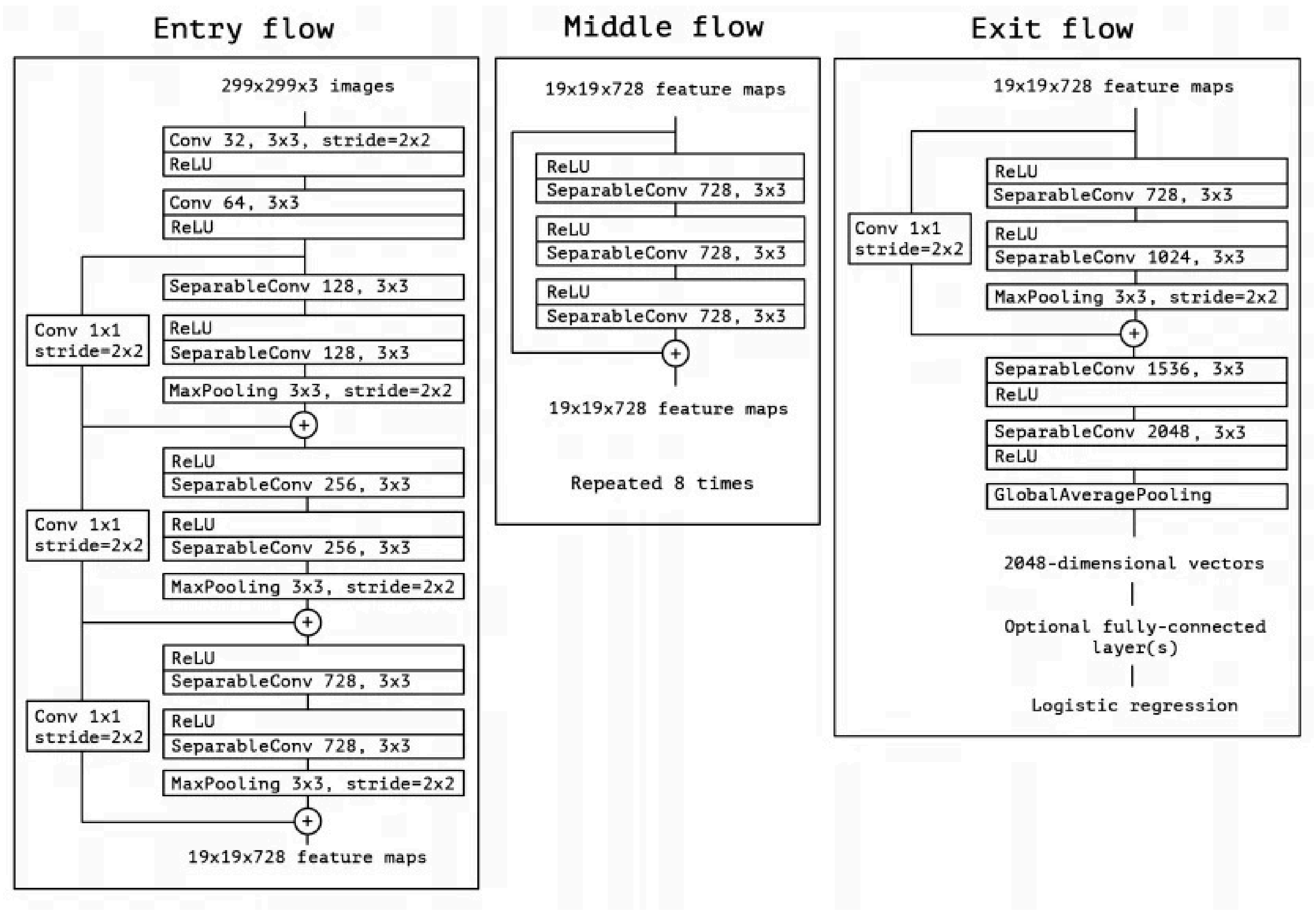
THE MODEL USED ON OUR DATASET

- **Xception [Xception: Deep Learning with Depthwise Separable Convolutions"https://arxiv.org/abs/1610.02357"]**
- **RESNET(18,34,50,101,152)[Deep Residual Learning for Image Recognition"https://ieeexplore.ieee.org/document/7780459"]**
- **DENSENET121[DENSELY CONNECTED CONVOLUTIONAL NETWORKS"HTTPS://ARXIV.ORG/ABS/1608.06993"]**

Xception :

We present an interpretation of Inception modules in convolutional neural networks as being an intermediate step in-between regular convolution and the depthwise separable convolution operation (a depthwise convolution followed by a pointwise convolution). In this light, a depthwise separable convolution can be understood as an Inception module with a maximally large number of towers. This observation leads us to propose a novel deep convolutional neural network architecture inspired by Inception, where Inception modules have been replaced with depthwise separable convolutions. We show that this architecture, dubbed Xception, slightly outperforms Inception V3 on the ImageNet dataset (which Inception V3 was designed for), and significantly outperforms Inception V3 on a larger image classification dataset comprising 350 million images and 17,000 classes. Since the Xception architecture has the same number of parameters as Inception V3, the performance gains are not due to increased capacity but rather to a more efficient use of model parameters.

model architecture:



1.Entry Flow:

- Input: The input to the Xception network is typically an image of size 299x299x3 (height x width x RGB channels).
- Initial Convolutions: The entry flow starts with a few standard convolutional layers (not depthwise separable) to perform initial feature extraction.
- Max Pooling: Max pooling layers are used to downsample the spatial dimensions, reducing the computational load.
  - Structure:Conv layer (3x3, 32 filters, stride 2)
  - Batch Normalization
  - ReLU activation
  - Conv layer (3x3, 64 filters)
  - Batch Normalization
  - ReLU activation
  - Max Pooling (3x3, stride 2)
  - Residual connection starts here

2. Middle Flow:
   - Core of the Network: This section is the heart of the Xception architecture, consisting of 8 identical blocks stacked sequentially.
   - Residual Connections: Each block has a residual connection, where the input to the block is added to its output. This helps with gradient flow during training, enabling the network to learn more effectively.
     - Structure of each block:ReLU activation
     - Depthwise Conv (3x3)
     - Batch Normalization
     - ReLU activation
     - Depthwise Conv (3x3)
     - Batch Normalization
     - ReLU activation
     - Depthwise Conv (3x3)
     - Batch Normalization
     - Residual connection (addition of the initial block input to the output)
3. Exit Flow:
   - Final Processing: This section performs the final processing of the extracted features before classification.
     - Structure:ReLU activation
     - Depthwise Conv (3x3)
     - Batch Normalization
     - ReLU activation
     - Depthwise Conv (3x3)
     - Batch Normalization
     - Global Average Pooling: This reduces the spatial dimensions to 1x1, resulting in a feature vector.
     - Fully Connected Layer (Dense layer): This layer performs the final classification.
     - Softmax activation: Outputs the probability distribution over the classes.

Key Architectural Features:

- Depthwise Separable Convolutions: As discussed earlier, these are the key building blocks of Xception, making it efficient.
- Residual Connections: These connections help with training deep networks by allowing gradients to flow more easily.
- Linear Stack: Xception has a relatively simple and linear structure compared to Inception, making it easier to implement and understand.

The key advantage:

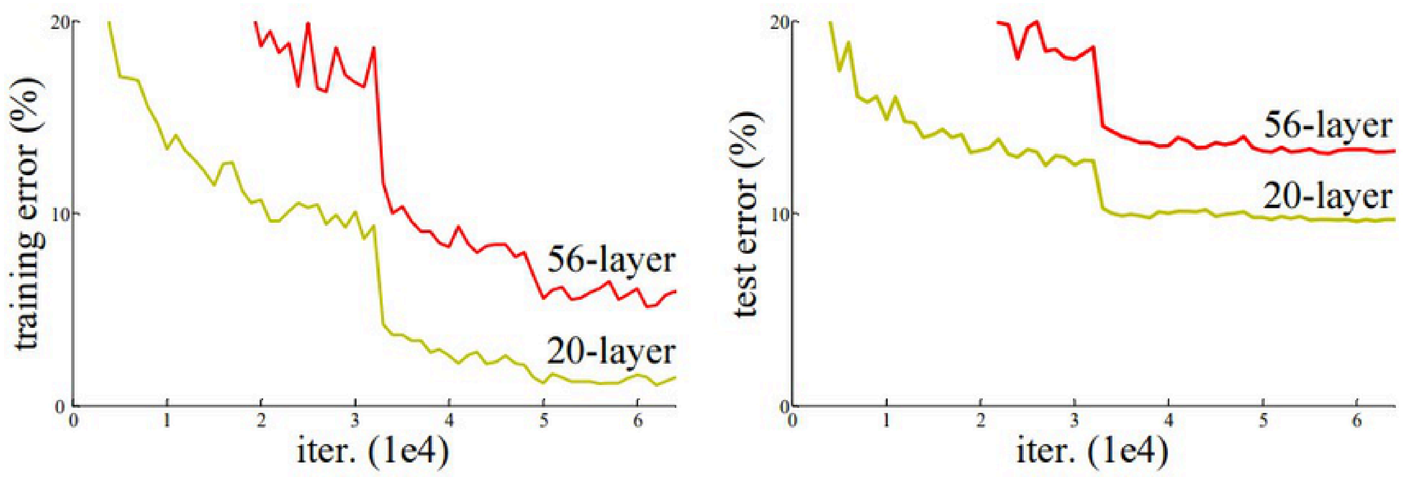
- Reduced Computational Cost: Depthwise separable convolutions significantly reduce the number of multiplications and additions required during both training and inference. This makes the model faster to train and run, especially on devices with limited processing power like mobile phones or embedded systems.
- Fewer Parameters: By factorizing standard convolutions, Xception uses far fewer parameters compared to models with traditional convolutions (like some earlier Inception versions or VGG). This has several benefits:
- Reduced Memory Footprint: Fewer parameters mean the model requires less memory to store, which is crucial for deployment on resource-constrained devices.
- Less Prone to Overfitting: With fewer parameters, the model has a lower capacity to memorize the training data, making it less likely to overfit, especially when dealing with smaller datasets.
- Strong Performance: Despite its efficiency, Xception often achieves comparable or even better performance than its predecessor, Inception V3, especially on larger datasets. This demonstrates that the factorization into depthwise and pointwise convolutions doesn't come at the cost of accuracy.

Limitations of xception:

- Performance Dependence on Dataset Size: Xception tends to perform exceptionally well on large datasets where it can fully leverage its capacity. However, on smaller datasets, it might be more prone to overfitting compared to simpler architectures. In such cases, strong regularization techniques or data augmentation become crucial.
- Not Always Superior to Other Architectures: While Xception often outperforms Inception V3, it doesn't necessarily outperform all other architectures in every scenario. The optimal choice of architecture often depends on the specific task, dataset, and computational resources available. More recent architectures like EfficientNet often achieve better performance with even fewer resources.

RESNET:

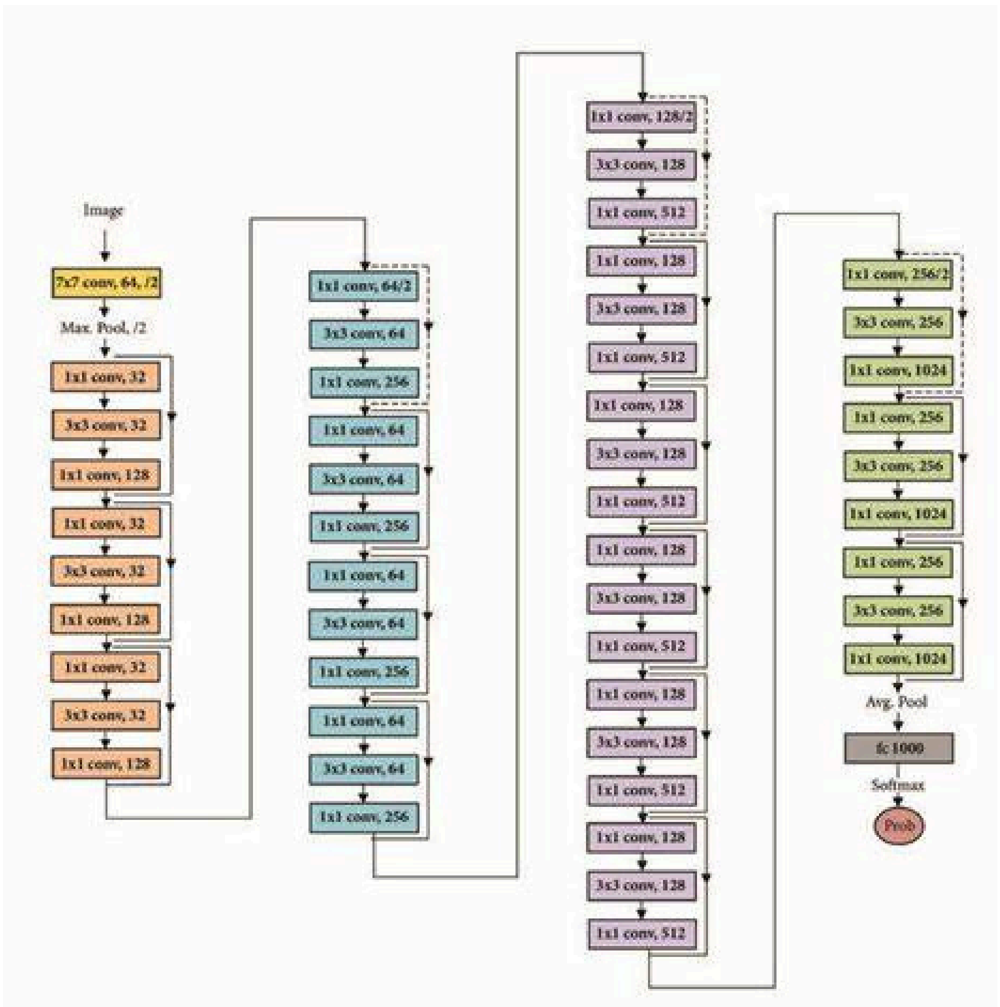
After the first CNN-based architecture (AlexNet) that win the ImageNet 2012 competition, Every subsequent winning architecture uses more layers in a deep neural network to reduce the error rate. This works for less number of layers, but when we increase the number of layers, there is a common problem in deep learning associated with that called the Vanishing/Exploding gradient. This causes the gradient to become 0 or too large. Thus when we increases number of layers, the training and test error rate also increases.



In the above plot, we can observe that a 56-layer CNN gives more error rate on both training and testing dataset than a 20-layer CNN architecture. After analyzing more on error rate the authors were able to reach conclusion that it is caused by vanishing/exploding gradient.
ResNet, which was proposed in 2015 by researchers at Microsoft Research introduced a new architecture called Residual Network.

Residual Network: In order to solve the problem of the vanishing/exploding gradient, this architecture introduced the concept called Residual Blocks. In this network, we use a technique called skip connections. The skip connection connects activations of a layer to further layers by skipping some layers in between. This forms a residual block. Resnets are made by stacking these residual blocks together.

model architecture:



ResNet architectures are built by stacking multiple residual blocks. There are two main types of residual blocks:

- Basic Block: Used for shallower ResNets (e.g., ResNet-18, ResNet-34). It consists of two 3x3 convolutional layers with batch normalization and ReLU activation functions, along with a shortcut connection.
- Bottleneck Block: Used for deeper ResNets (e.g., ResNet-50, ResNet-101, ResNet-152). It uses three convolutional layers: a 1x1 convolution to reduce the number of channels, a 3x3 convolution, and another 1x1 convolution to increase the number of channels back. This "bottleneck" design reduces the computational cost.

Overall Architecture:

A typical ResNet architecture consists of:

Initial Convolution and Pooling: A convolutional layer followed by max pooling to perform initial feature extraction and downsampling.

Stacked Residual Blocks: Multiple stages of residual blocks are stacked together. Each stage typically has a different number of blocks and a different number of filters.

Global Average Pooling: This reduces the spatial dimensions to a single value per feature map.

Fully Connected Layer: A fully connected layer with a softmax activation function for classification.

The key advantage:

- Enables training of very deep networks: ResNet overcomes the vanishing gradient problem, allowing for the training of networks with hundreds or even thousands of layers.
- Improved accuracy: Deeper ResNets generally achieve higher accuracy than shallower networks.
- Ease of optimization: Residual connections make it easier to optimize the network during training.

Different ResNet Versions:

ResNet comes in different depths, such as ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, where the number indicates the number of layers in the network. Deeper ResNets generally have more residual blocks.

Limitations of RESNet:

- Still Computationally Intensive: Although residual connections mitigate the vanishing gradient problem and allow for deeper networks, very deep ResNets (like ResNet-152) can still be computationally expensive, requiring significant memory and processing power. This can be a limitation for deployment on resource-constrained devices.

- Diminishing Returns with Extreme Depth: While increasing depth initially improves performance, there are diminishing returns. Simply stacking more and more residual blocks doesn't always lead to significant gains in accuracy and can even lead to overfitting or other optimization challenges.

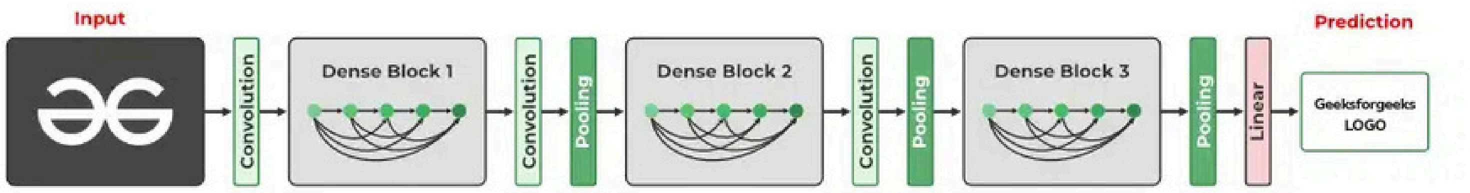
- Information Bottleneck: In some ResNet implementations, particularly those using bottleneck blocks, the 1x1 convolutions can create an information bottleneck, potentially limiting the flow of information within the network.

- Vanishing/Exploding Gradients Not Fully Eliminated: While ResNet significantly alleviates the vanishing gradient problem, it doesn't entirely eliminate it. In extremely deep networks or with certain training settings, gradient issues can still arise, though to a much lesser extent than in plain networks.

DENSENET :

DenseNet, short for Dense Convolutional Network, is a deep learning architecture for convolutional neural networks (CNNs) introduced by Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger in their paper titled "Densely Connected Convolutional Networks" published in 2017. DenseNet revolutionized the field of computer vision by proposing a novel connectivity pattern within CNNs, addressing challenges such as feature reuse, vanishing gradients, and parameter efficiency. Unlike traditional CNN architectures where each layer is connected only to subsequent layers, DenseNet establishes direct connections between all layers within a block. This dense connectivity enables each layer to receive feature maps from all preceding layers as inputs, fostering extensive information flow throughout the network.

model architecture:



DenseNet introduces a paradigm shift by connecting each layer to every other layer in a feed-forward manner. Unlike traditional CNNs, which have a single connection between consecutive layers, DenseNet ensures that each layer receives inputs from all preceding layers and passes its output to all subsequent layers. This results in a network with L(L+1)/2 direct connections for L layers, significantly enhancing information flow.

Dense blocks: are the building blocks of DenseNet architectures. Each dense block consists of multiple convolutional layers, typically followed by batch normalization and a non-linear activation function (e.g., ReLU). Importantly, each layer within a dense block receives feature maps from all preceding layers as inputs, facilitating feature reuse and propagation.

Within a dense block, each layer receives the concatenated output of all preceding layers as its input. If a dense block has m layers, and each layer produces k feature maps (where k is known as the growth rate), the l-th layer will have k×(l+l0)$k×(l+l0)$ input feature maps (where l0$l0$ is the number of input channels to the dense block).

Example of a Dense Block:

1. Layer 1: Receives input feature maps.
2. Layer 2: Receives input feature maps + output of Layer 1.
3. Layer 3: Receives input feature maps + output of Layer 1 + output of Layer 2.

This pattern continues for all layers within the block, ensuring a highly interconnected architecture.

Transition Layer

Transition layers are used to connect dense blocks. They serve two main purposes: reducing the number of feature maps and downsampling the spatial dimensions of the feature maps. This helps in maintaining the computational efficiency and compactness of the network. A typical transition layer consists of:

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

Growth Rate (k)

The growth rate (k) is a critical hyperparameter in DenseNet. It defines the number of feature maps each layer in a dense block produces. A larger growth rate means more information is added at each layer, but it also increases the computational cost. The choice of k affects the network's capacity and performance.

DenseNet Variants:

DenseNet comes in several variants, distinguished primarily by their depth and number of layers:

DenseNet-121: Contains 121 layers, known for its balanced trade-off between computational efficiency and accuracy. Ideal for tasks requiring moderate computational resources.

DenseNet-169: With 169 layers, this variant provides deeper feature extraction, suitable for more complex datasets where higher accuracy is needed.

DenseNet-201 and DenseNet-264: These variants offer even deeper architectures, suitable for highly complex tasks requiring extensive feature representation.

The key advantage:

- Reduced Vanishing Gradient Problem: Dense connections improve gradient flow and facilitate the training of very deep networks.
- Feature Reuse: Each layer has access to all preceding layers' feature maps, promoting the reuse of learned features and enhancing learning efficiency.
- Fewer Parameters: DenseNets often have fewer parameters compared to traditional CNNs with similar depth due to efficient feature reuse.
- Improved Accuracy: DenseNets have shown high accuracy on various benchmarks, such as ImageNet and CIFAR.

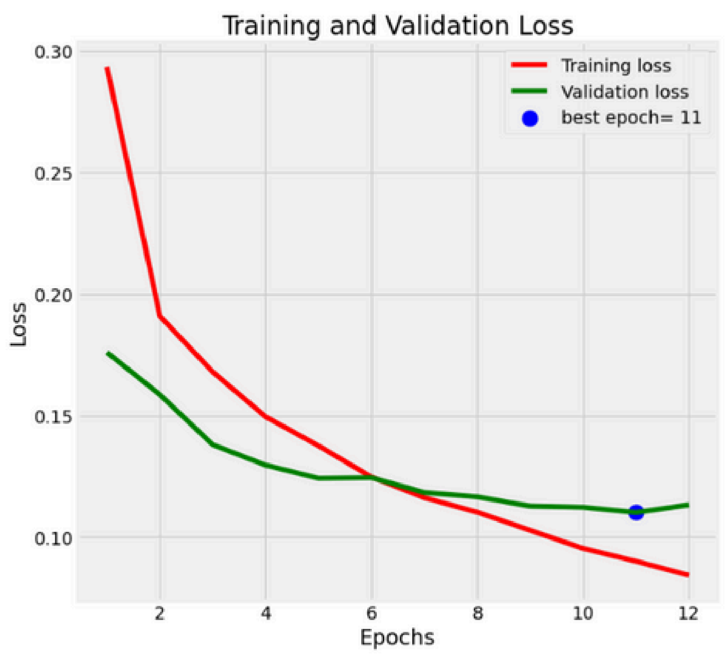
Limitations of DenseNet:

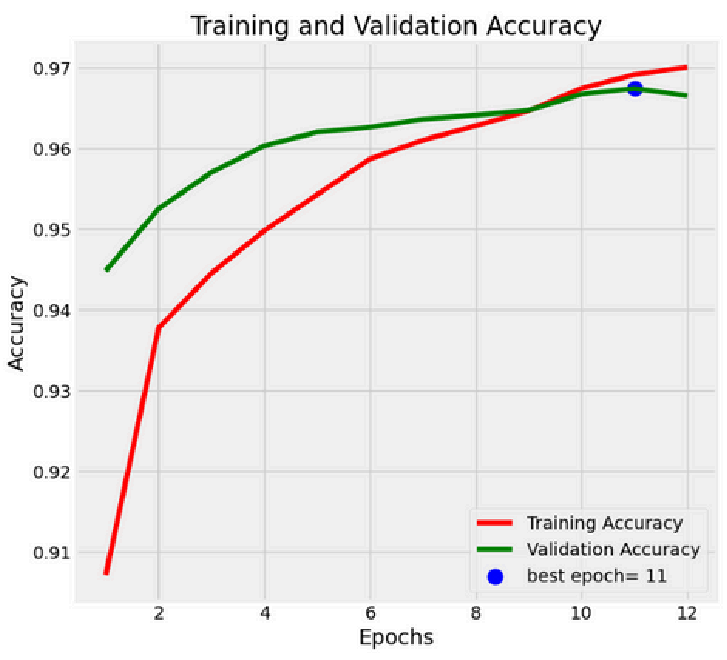
- High Memory Consumption: Dense connections increase memory usage due to the storage requirements for feature maps, making DenseNet less practical for devices with limited memory.
- Computational Complexity: The extensive connectivity leads to increased computational demands, resulting in longer training times and higher computational costs, which may not be ideal for real-time applications.
- Implementation Complexity: Managing and concatenating a large number of feature maps adds complexity to the implementation, requiring careful tuning of hyperparameters and regularization techniques to maintain performance and stability.
- Risk of Overfitting: Although DenseNet reduces overfitting through better feature reuse, there is still a risk, particularly if the network is not properly regularized or if the training data is insufficient.

PERFORMANCE OF THE MODELS ON OUR DATASET

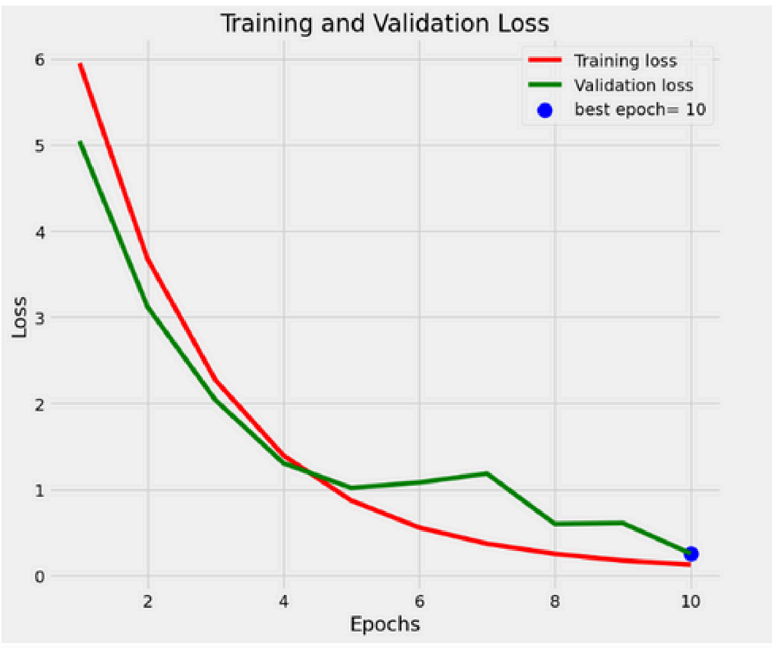
TRAINING AND VALIDATION ACCURACY GRAPH:

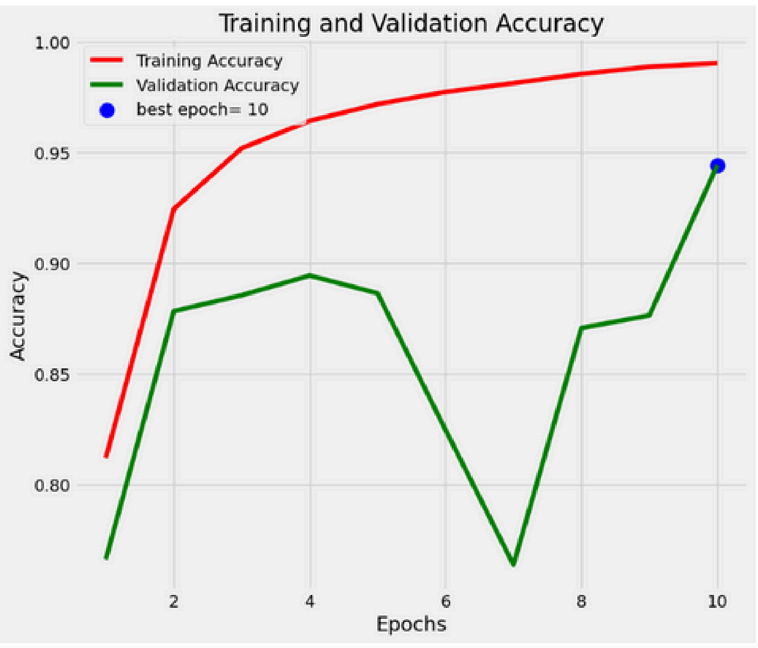
1-XCEPTION MODEL:



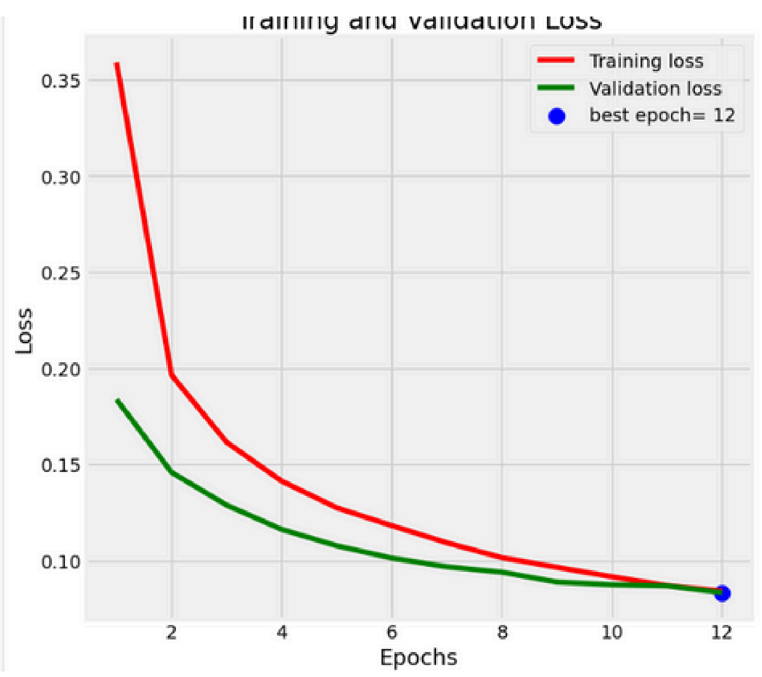


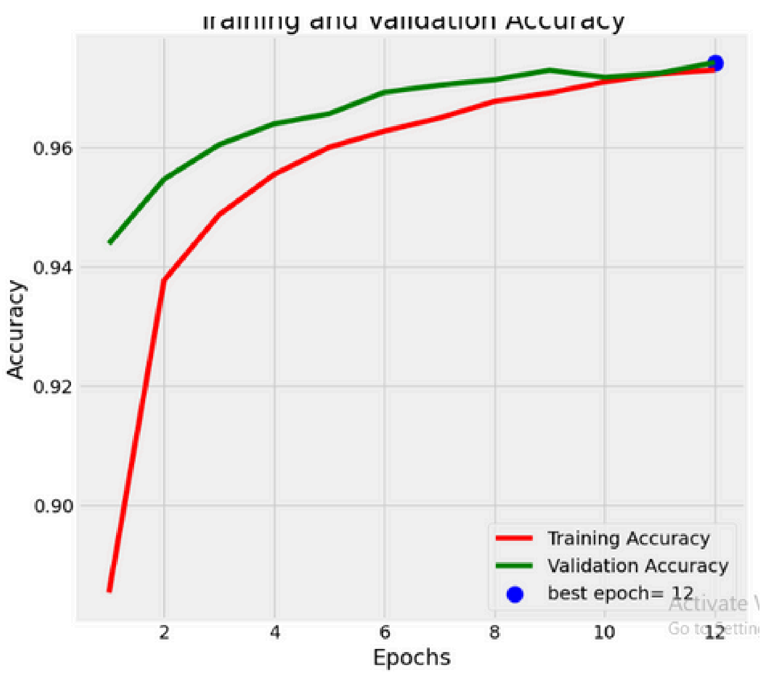
2-RESNET MODEL:





3-DESNET MODEL:



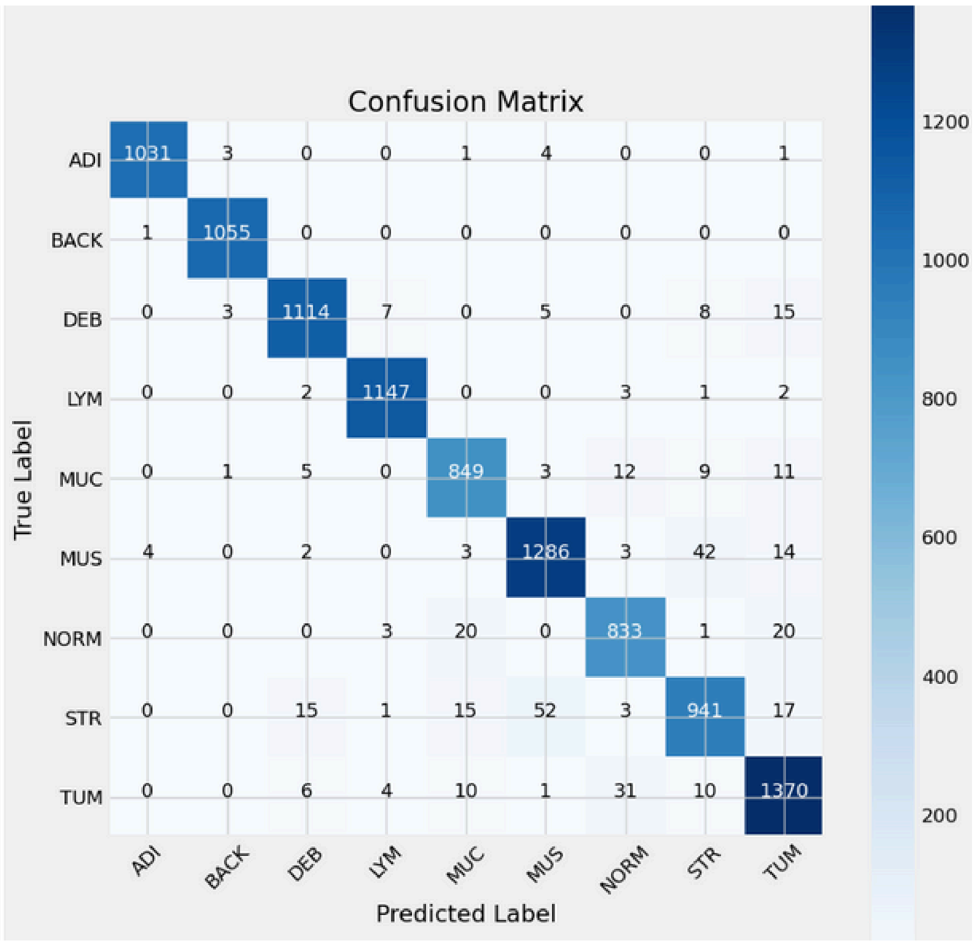


LOSS AND ACCURACY:

MODEL	Train Loss	Train Accuracy	Valid Loss	Valid Accuracy	Test Loss	Test Accuracy
ResNet	0.188	0.961	0.253	0.944	0.256	0.946
Xception	0.047	0.984	0.110	0.967	0.124	0.962
DenseNet	0.061	0.980	0.083	0.974	0.090	0.970

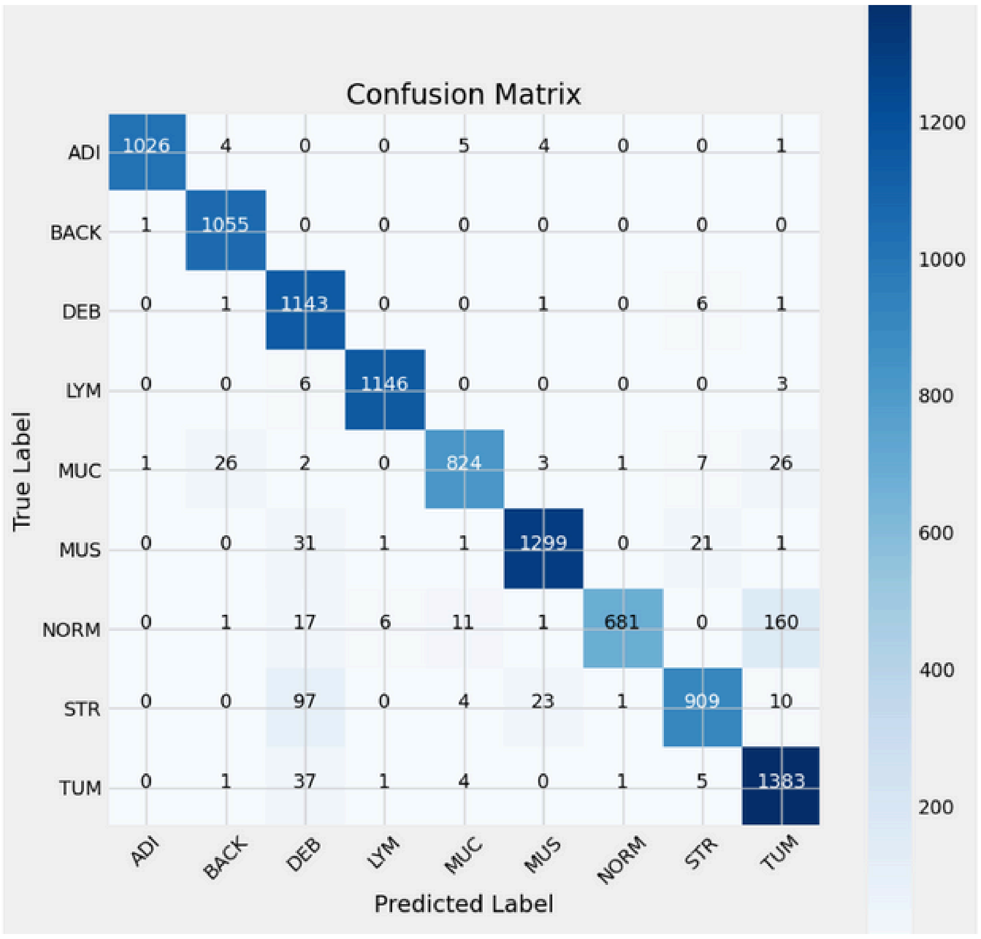
CONFUSION MATRIX:

1-XCEPTION MODEL:



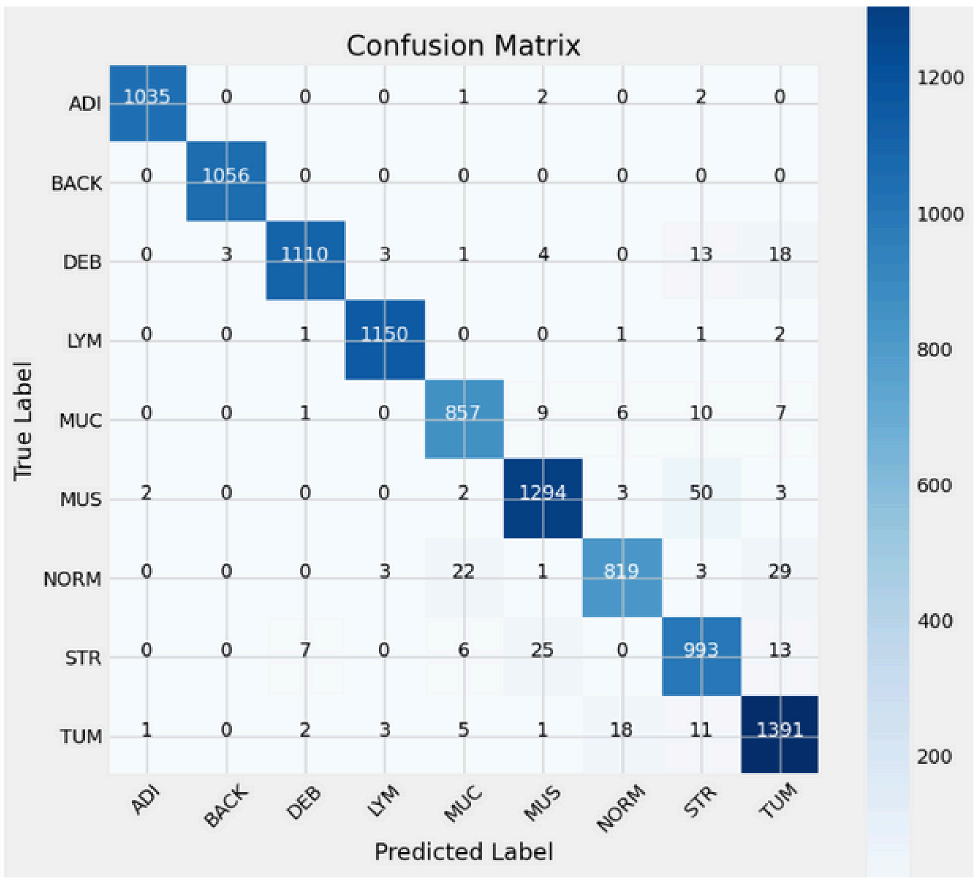
True Label \ Predicted Label	ADI	BACK	DEB	LYM	MUC	MUS	NORM	STR	TUM
ADI	1031	3	0	0	1	4	0	0	1
BACK	1	1055	0	0	0	0	0	0	0
DEB	0	3	1114	7	0	5	0	8	15
LYM	0	0	2	1147	0	0	3	1	2
MUC	0	1	5	0	849	3	12	9	11
MUS	4	0	2	0	3	1286	3	42	14
NORM	0	0	0	3	20	0	833	1	20
STR	0	0	15	1	15	52	3	941	17
TUM	0	0	6	4	10	1	31	10	1370

2-RESNET MODEL:



True Label \ Predicted Label	ADI	BACK	DEB	LYM	MUC	MUS	NORM	STR	TUM
ADI	1026	4	0	0	5	4	0	0	1
BACK	1	1055	0	0	0	0	0	0	0
DEB	0	1	1143	0	0	1	0	6	1
LYM	0	0	6	1146	0	0	0	0	3
MUC	1	26	2	0	824	3	1	7	26
MUS	0	0	31	1	1	1299	0	21	1
NORM	0	1	17	6	11	1	681	0	160
STR	0	0	97	0	4	23	1	909	10
TUM	0	1	37	1	4	0	1	5	1383

3-DESNET MODEL:

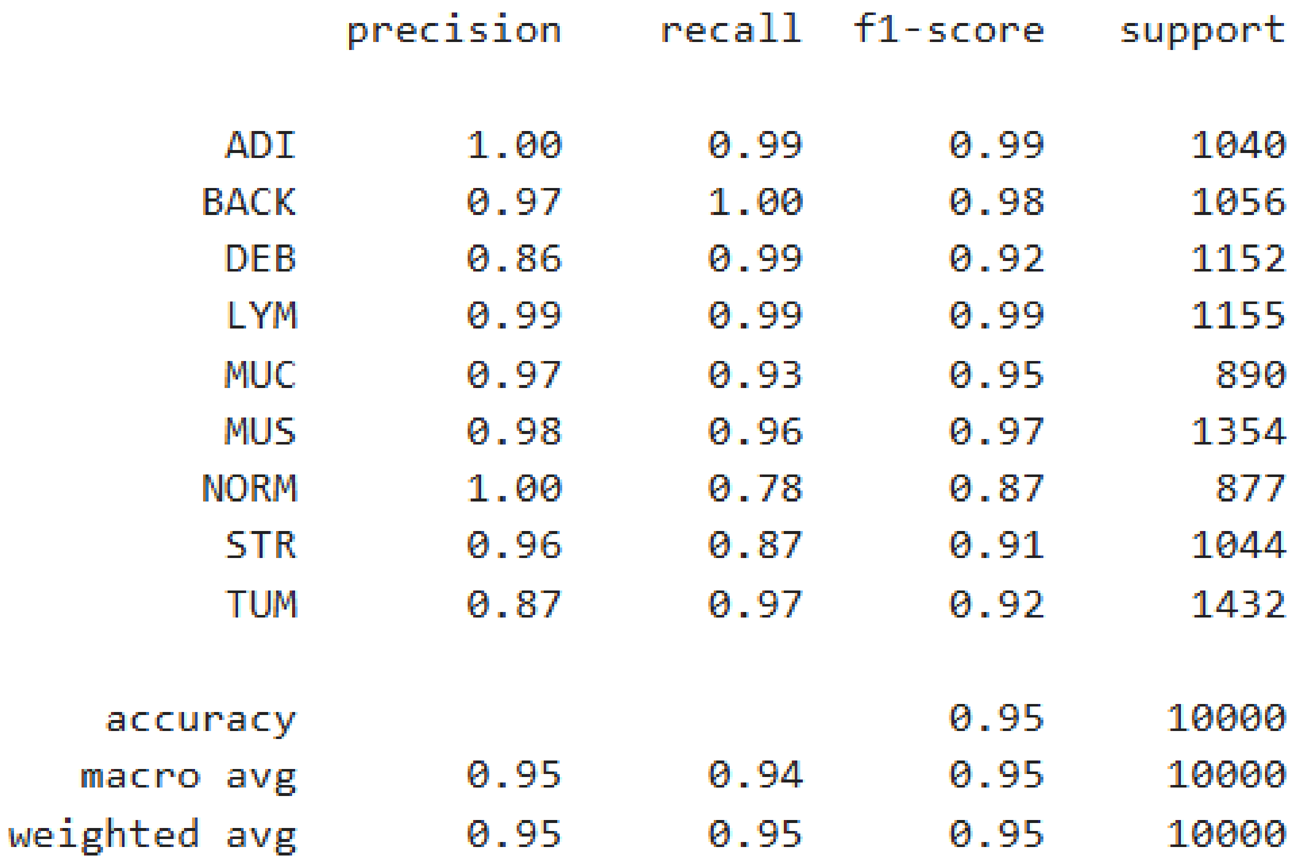


CLASSIFICATION_REPORT:

1-XCEPTION MODEL:

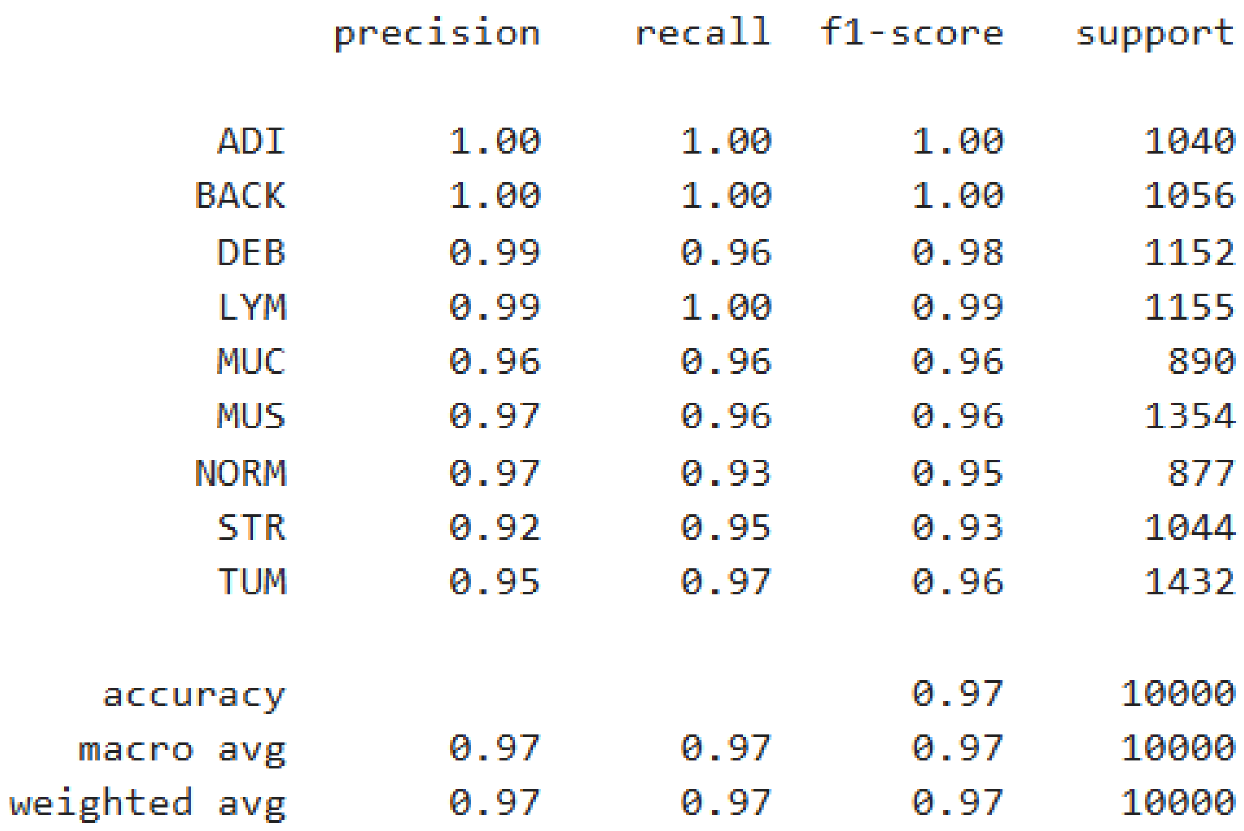
```
              precision    recall  f1-score   support

         ADI       1.00      0.99      0.99      1040
        BACK       0.99      1.00      1.00      1056
         DEB       0.97      0.97      0.97      1152
         LYM       0.99      0.99      0.99      1155
         MUC       0.95      0.95      0.95       890
         MUS       0.95      0.95      0.95      1354
        NORM       0.94      0.95      0.95       877
         STR       0.93      0.90      0.92      1044
         TUM       0.94      0.96      0.95      1432

    accuracy                           0.96     10000
   macro avg       0.96      0.96      0.96     10000
weighted avg       0.96      0.96      0.96     10000
```

2-RESNET MODEL:



```
              precision    recall  f1-score   support

         ADI       1.00      0.99      0.99      1040
        BACK       0.97      1.00      0.98      1056
         DEB       0.86      0.99      0.92      1152
         LYM       0.99      0.99      0.99      1155
         MUC       0.97      0.93      0.95       890
         MUS       0.98      0.96      0.97      1354
        NORM       1.00      0.78      0.87       877
         STR       0.96      0.87      0.91      1044
         TUM       0.87      0.97      0.92      1432

    accuracy                           0.95     10000
   macro avg       0.95      0.94      0.95     10000
weighted avg       0.95      0.95      0.95     10000
```

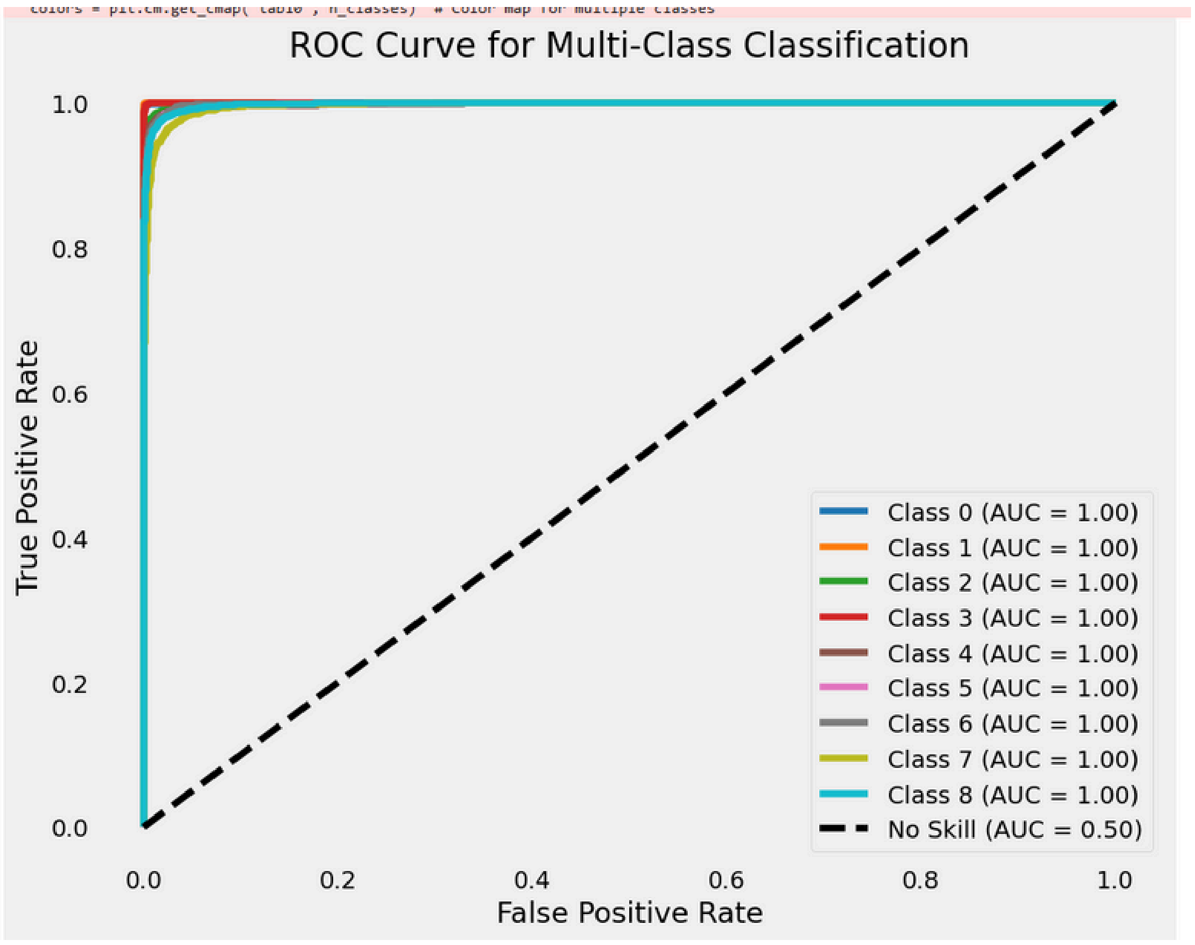
3-DESNET MODEL:



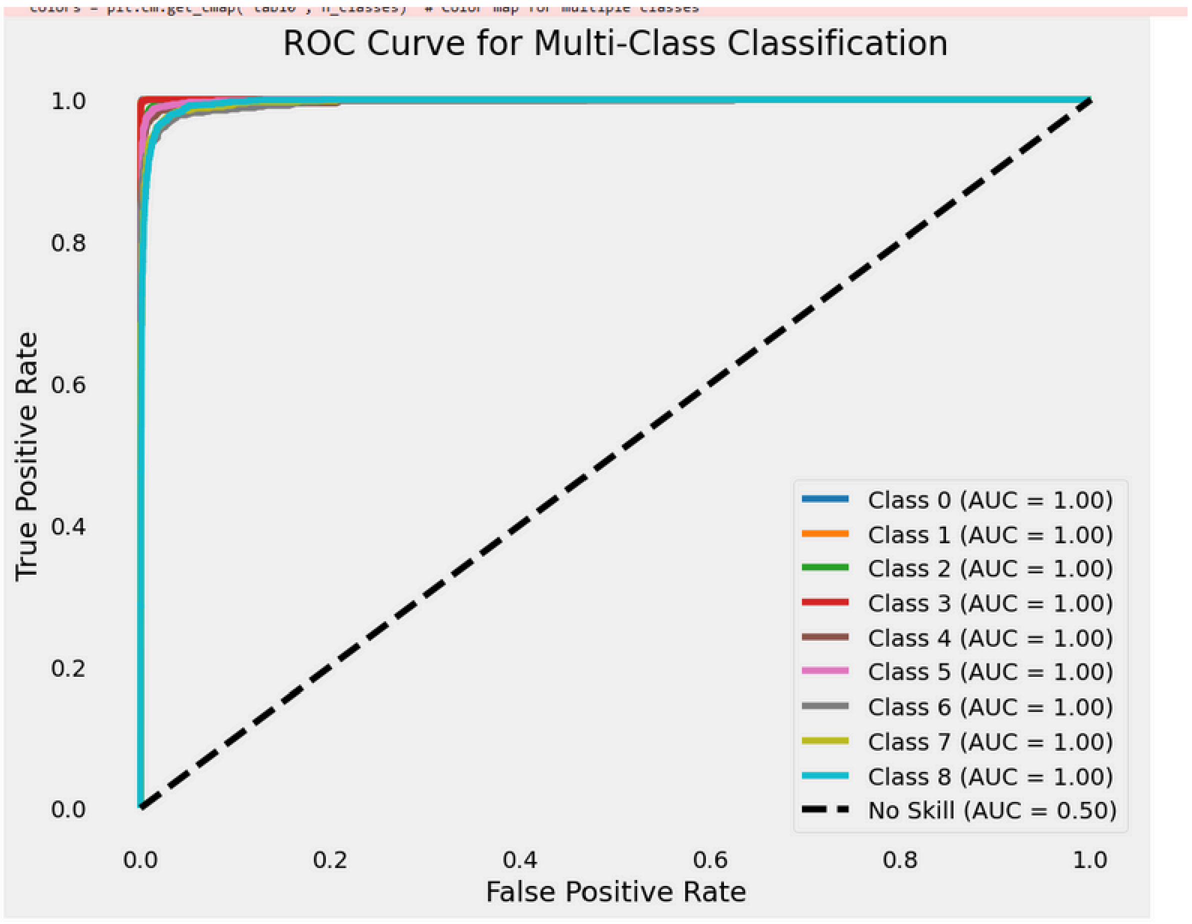
```
              precision    recall  f1-score   support

         ADI       1.00      1.00      1.00      1040
        BACK       1.00      1.00      1.00      1056
         DEB       0.99      0.96      0.98      1152
         LYM       0.99      1.00      0.99      1155
         MUC       0.96      0.96      0.96       890
         MUS       0.97      0.96      0.96      1354
        NORM       0.97      0.93      0.95       877
         STR       0.92      0.95      0.93      1044
         TUM       0.95      0.97      0.96      1432

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

ROC CURVE FOR MULTI-CLASS CLASSIFICATION:

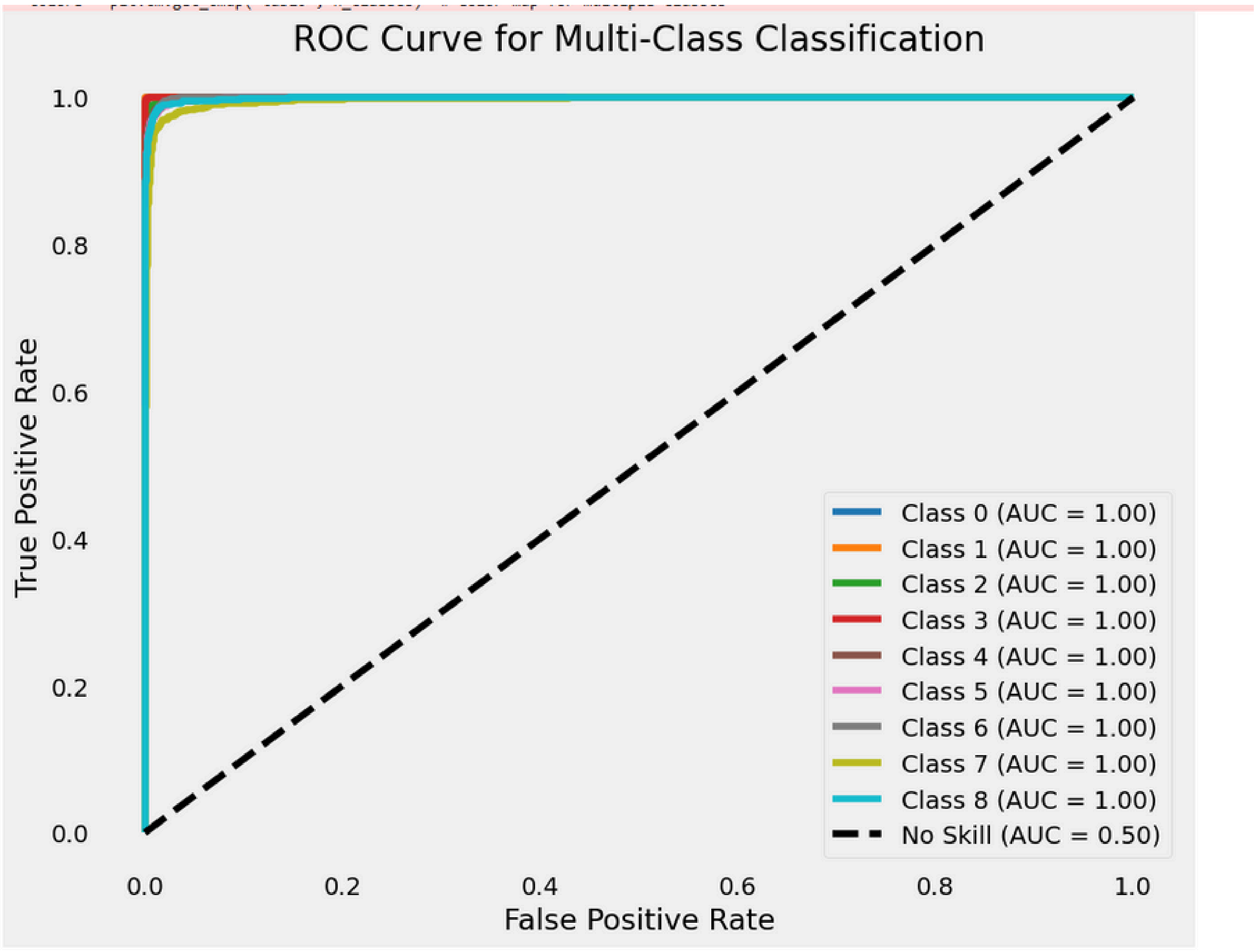
1-XCEPTION MODEL:



2-RESNET MODEL:



3-DESNET MODEL:



ARCHITECTURE OVERVIEW

Aspect	DenseNet121	Xception	ResNet
Core Idea	Dense connections between layers to maximize feature reuse and improve gradient flow.	Depthwise separable convolutions for efficient computation.	Residual connections to mitigate vanishing gradient issues.
Parameter Efficiency	Highly parameter-efficient due to shared feature maps.	Moderately efficient with lightweight depthwise separable convolutions.	Higher parameter count compared to DenseNet.
Model Depth	121 layers (densely connected blocks).	36 convolutional layers (lightweight).	50+ layers with residual blocks.
Computation Cost	Higher due to dense connections.	Lower due to separable convolutions.	Moderate, depending on depth.

SUITABILITY FOR OUR DATASET

- **DenseNet121**: Best suited for extracting fine-grained features in histopathology images, particularly for high-resolution data with subtle texture differences.
- **Xception**: Ideal when computational resources are limited, offering decent performance with lower hardware demands.
- **ResNet**: A balanced choice, offering strong performance without the memory overhead of DenseNet.

PROS AND CONS FOR EACH MODEL

1-XCEPTION MODEL:

Pros:

- **Computational Efficiency**: Depthwise separable convolutions significantly reduce the number of parameters and computations, making it faster to train and deploy.
- **Lightweight**: Performs well on hardware with limited computational power.
- Good Performance: Delivers competitive accuracy despite being lighter.

Cons:

- **Limited Feature Extraction**: May struggle to capture complex, fine-grained features in high-resolution histopathology images.
- **Lower Robustness**: Slightly more prone to underfitting on small datasets with diverse texture variations.

2-RESNET MODEL:

Pros:

- **Residual Connections**: Makes training deep networks easier by addressing vanishing gradient problems.
- **Balanced Architecture**: Combines efficiency and performance, making it a reliable baseline.
- **Scalability**: Easily adapted to different model sizes (e.g., ResNet50, ResNet101).

Cons:

- **Higher Parameter Count**: Requires more memory than Xception and DenseNet.
- **Moderate Computational Cost**: Slower than Xception but faster than DenseNet.
- **Potential Redundancy**: May include unnecessary parameters compared to DenseNet.

3-DENSENET MODEL:

Pros:

- **Feature Reuse**: Each layer connects to every other layer, ensuring maximum feature reuse, which is crucial for capturing intricate textures in histopathology images.
- **Efficient Gradient Flow**: Direct connections reduce vanishing gradient problems, enhancing performance on smaller datasets like NCT-CRC-HE-100K.
- **Compact Representations**: Efficient use of parameters leads to a model that performs exceptionally well relative to its size.
- **Robustness**: Handles variability in histopathological data effectively.

Cons:

- **High Computational Cost**: Dense connections increase training time and memory usage.
- **Slower Inference**: Requires more resources, making it less ideal for real-time applications.

RECOMMENDATION

- Use **DenseNet121** for maximum accuracy if computational resources are not a constraint.
- Use **Xception** for resource-efficient training and inference with slight accuracy trade-offs.
- Use **ResNet** for a middle-ground approach that balances performance and efficiency.

BEST ARCHITECTURE FOR OUR DATA

- **DenseNet121** is particularly advantageous for the NCT-CRC-HE-100K dataset due to its unique properties:
- **Texture-Rich Data**: Histopathological images contain subtle texture variations that **DenseNet121** can capture effectively due to feature reuse and dense connectivity.
- **Small Dataset**: The dataset size (100K images) is relatively small for deep learning tasks, making DenseNet’s efficient gradient flow and compact representations a significant advantage.

- **Fine-Grained Classification**: The detailed feature extraction capability of DenseNet121 ensures that subtle differences in cell morphology are identified, improving classification performance.
- **Robustness**: The model is less prone to overfitting, which is crucial for datasets with high inter-class variability, such as cancer subtypes.

references :

- https://paperswithcode.com/sota/medical-image-classification-on-nct-crc-he
- https://www.ibm.com/think/topics/transfer-learning
- https://www.geeksforgeeks.org/densenet-explained/
- https://keras.io/api/applications/xception/
- https://towardsdatascience.com/xception-from-scratch-using-tensorflow-even-better-than-inception-940fb231ced9
- https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/
- https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8
- https://www.geeksforgeeks.org/densenet-explained/
- https://www.scielo.br/j/cta/a/DS7zdcncYx9QyL6tRQxpRhH/?lang=en
- https://www.run.ai/guides/deep-learning-for-computer-vision/pytorch-resnet#:~:text=Running%20ResNet%20on%20PyTorch%20with,computing%20intensive%20neural%20network%20architecture.
-