**Dense Convolutional Network (DenseNet) on CIFAR-10 and the CIFAR-100**

**Yangyu Chen**

**Abstract:**

This report investigates the effectiveness of Dense Convolutional Networks (DenseNet) on CIFAR-10 and CIFAR-100 for image classification tasks. By leveraging dense connectivity and optimization techniques, DenseNet demonstrates high accuracy and computational efficiency, outperforming Vision Transformers (ViT) on small datasets. These findings provide insights into model selection for resource-constrained image classification applications.

**Keywords:** deep learning;Densest; CIFAR-10; neural networks; Vision transformer.

**Introduction(**Github link**: https://github.com/cyy200107/DS-Final-project.git):**

Deep learning has revolutionized computer vision tasks, with Convolutional Neural Networks (CNNs) playing a pivotal role in achieving state-of-the-art performance in image classification, object detection, and segmentation. Among these architectures, Dense Convolutional Networks (DenseNet) stand out due to their unique design, which promotes feature reuse and alleviates the vanishing gradient problem through dense connectivity.

This study investigates the performance of DenseNet on the CIFAR-10 and CIFAR-100 datasets, focusing on three primary objectives:

1. To optimize DenseNet for small datasets by fine-tuning its architecture and training process.
2. To compare DenseNet with Vision Transformers (ViT) to understand their respective strengths and weaknesses.

3. To analyze DenseNet's computational efficiency and scalability for real-world applications.

The report highlights the advantages of DenseNet's design, such as bottleneck and transition layers, and demonstrates its adaptability to limited-resource scenarios. Furthermore, this study provides a baseline for future research into hybrid architectures and transformer-based networks for small-scale image classification tasks.

**Related Work**

Convolutional Neural Networks (CNNs) have revolutionized the computer vision(cv). CNNs achieved state-of-the-art (Sota) performance in cv tasks (before the appear vit and other transformer-based method) such as image classification, object detection, and semantic segmentation. Among these architectures, Dense Convolutional Networks (DenseNets)[1] have gained significant attention due to its design that promotes feature reuse and improves gradient flow. This report details the application of DenseNet on the CIFAR-10 dataset and the CIFAR-100 dataset, proving the architecture’s effectiveness and the methodologies employed during the training and evaluation stages.

**Convolutional Neural Networks (CNNs)**

CNNs are neural networks specifically designed to process grid-like data by **Convolution**, such as images. They consist of convolutional layers that use learnable filters across the input space to detect local dependencies and spatial features of data. Traditional CNNs suffer from vanishing gradient problems when they become too deep, making it challenging to train very deep networks effectively. The invention of

residual block(resnet) figures the problem, making it possible to train very deep neural networks. DenseNet borrows this idea.

# Methodology

### DenseNet Architecture

DenseNets introduce a paradigm where each layer is directly connected to every other layer in a feed-forward form. This design ensures that features learned at each layer are available to all layers, promoting feature reuse and alleviating the vanishing gradient problem. Additionally, DenseNets are computationally efficient due to the bottleneck layers that reduce the dimensionality of the feature map.

### DenseNet Design Principles

### Key Components

### Bottleneck Layers

Bottleneck layers are important design of DenseNet that help reduce the computational complexity. They perform a 1×1 convolution to decrease the number of input channels, followed by a 3×3 convolution to extract features. This process allows for efficient feature extraction.

### Transition Layers

Transition layer downsample the feature maps and reduce the number of channels. Batch normalization, a 1×1 convolution, and an average pooling operation are used.

### Vision Transformer

Vision Transformer [4](ViT) is a deep learning model based on the Transformer architecture, primarily used for image recognition and other computer vision tasks. Unlike traditional Convolutional Neural Networks (CNNs), ViT treats images as serialized inputs and leverages self-attention mechanisms to process relationships between pixels in the image. ViT first divides the input image into fixed-size patches,

such as 16x16 pixels, and then flattens each patch into a one-dimensional vector. These vectors are mapped into a high-dimensional space through a linear projection layer, forming the embedded representation of the image patches. Since the Transformer model itself does not have the ability to capture sequence order, ViT adds positional embeddings to each image patch to maintain spatial structure information. The core of ViT is multiple Transformer encoder layers, each of which includes multi-head self-attention and a feed-forward neural network. These encoder layers process the embedded vectors of the image patches, calculate global dependencies between image patches, and gradually extract global features. ViT introduces a special classification token (CLS Token) for the final classification task. The CLS Token is input into the Transformer along with the embedded vectors of other image patches and represents the global features of the entire image in the final output. Finally, the CLS Token is sent to a fully connected layer for classification. After processing, the CLS Token output from ViT is passed through a fully connected layer to generate the final classification results. ViT's advantage lies in its ability to capture the global context of the entire image, which is very helpful for fine-grained segmentation tasks. It can effectively distinguish between visually similar but semantically different objects, improving segmentation accuracy. ViT also performs well in image generation and reconstruction tasks, where its self-attention mechanism can model globally and generate more realistic image content. Overall, ViT achieves feature extraction and classification of images by dividing images into small patches, treating them as serialized tokens, and inputting them into the Transformer encoder, a process similar to the Transformer model used for processing text sequences in natural language processing.

## Experiments

### DATA Preprocessing

Before training the model, to improve the network's performance. the images were normalized. This step normalizes the pixel values to have selected mean and unit variance. In deep learning, appropriate preprocessing of image data is crucial as it helps the model learn and generalize better. In this code, preprocessing steps include converting images to PIL images, applying random horizontal flipping, random rotation, converting to Tensors, and normalization. For testing data, the preprocessing steps are slightly simpler, including conversion to PIL images, conversion to Tensors, and normalization.Converting to PIL images is the first step in preprocessing because PyTorch's ToPILImage() transformation can convert image data from NumPy array format to PIL image format. PIL (Python Imaging Library) is an image processing library that provides a wealth of image manipulation features, facilitating subsequent processing.Random horizontal flipping and random rotation are data augmentation techniques that increase the diversity of the data by randomly flipping and rotating images. The benefit of this is to prevent the model from becoming overly dependent on specific directions or positions in the original dataset, thereby improving the model's generalization ability. For example, random horizontal flipping can ensure that the model is not sensitive to the left-right direction of the image, while random rotation can ensure that the model has some robustness to different orientations of the image.Converting to Tensors involves transforming PIL images into PyTorch's Tensor format because PyTorch models can only process data in Tensor format. This conversion step is necessary as it allows the image data to be read and processed by the model.Normalization scales the image data to a specific range (usually between 0 and 1) by subtracting the mean and dividing by the standard deviation. In this code,

the mean and standard deviation for each color channel are set to 0.5, which means that each pixel value of the image will be scaled to between -1 and 1. Normalization helps to speed up the convergence of the model during the training process because it ensures that different input features have the same scale, allowing the gradient descent algorithm to work more effectively.

**Model Initialization**

**Optimization Strategy**

The Adam[2] optimization algorithm was used.  The Adam optimization algorithm is an efficient optimization method used in deep learning, combining the advantages of Momentum and RMSprop algorithms. It adaptively adjusts the learning rate for each parameter by computing the first moment (mean) and second moment (uncentered variance) of the gradients. This approach is not only capable of dealing with sparse gradient issues but also suitable for large-scale data and parameter volume problems, as well as non-stationary objectives and very noisy or sparse gradient issues.The key advantage of the Adam algorithm lies in its adaptability, as it independently adjusts the learning rate for each parameter, making the algorithm more flexible when facing parameters with different gradient magnitudes. Additionally, Adam introduces a bias correction mechanism to address initialization bias issues, ensuring faster convergence in the early stages of training. Due to its low memory requirements and robustness to parameters, Adam has become one of the preferred optimizers in deep learning, especially suitable for complex non-convex optimization problems.

**Training Process**

The training phase involved feeding batches of images through the network, computing the loss based on the predicted output and the true label, and then adjusting the network weights via backpropagation. The training process was repeated for

several epochs until the network reached a satisfactory level of accuracy on the training data.

**Evaluation Metrics**

To assess the performance of the trained DenseNet model, the accuracy metric was used. Accuracy measures the proportion of correctly classified images out of the total number of images tested. The model was evaluated on a test set which was not seen during the training phase to provide an unbiased estimate of the model's generalization capabilities.

**Datasets**

The CIFAR-10 dataset is widely used. It consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. And the Humpback Whale Identification dataset contains thousands of images of humpback whale flukes. Individual whales have been identified by researchers and given an Id.

The CIFAR-100 dataset consists of 60,000 32×32 color images, divided into 100 classes. Each class in CIFAR-100 contains 600 images, making for a total of 6,000 images per class, just like in CIFAR-10. The major difference between CIFAR-10 and CIFAR-100 is the number of classes; CIFAR-100 has 10 times as many classes, but with fewer images per class. The images in CIFAR-100 are grouped into 20 superclasses, with each superclass containing 5 classes that share some visual or semantic relationship

**Results**

After training the DenseNet model on the CIFAR-10 dataset, the network demonstrated high accuracy on both the training and test sets. The model achieved a test accuracy of 89% after 3 epochs, indicating that the dense connectivity pattern facilitated effective learning and generalization. DenseNet has also demonstrated

promising results on the CIFAR-100 dataset. The loss and accuracy plots of DenseNet are shown below.

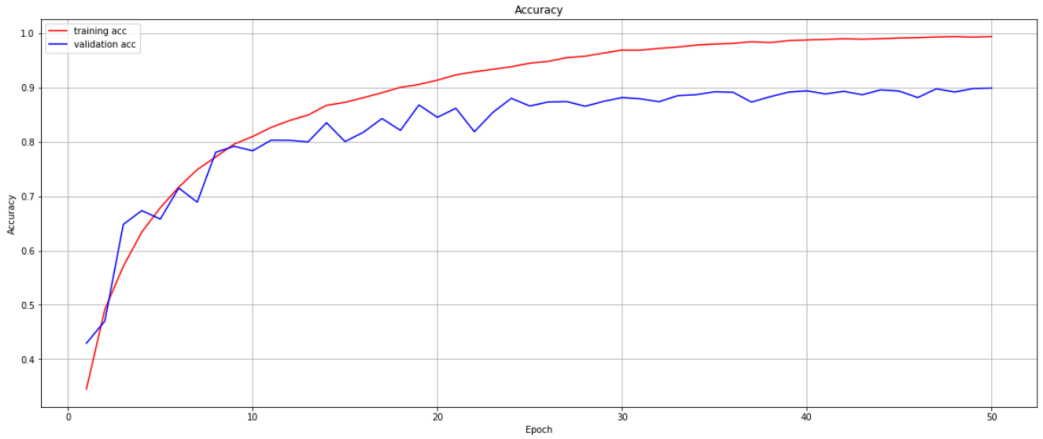


Figure 1: the accuracy of densenet

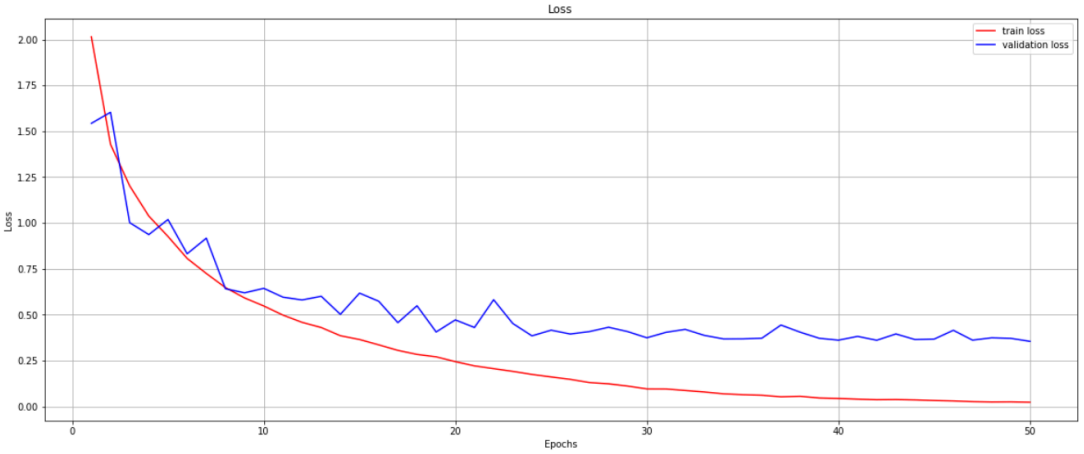


Figure 2: loss figure

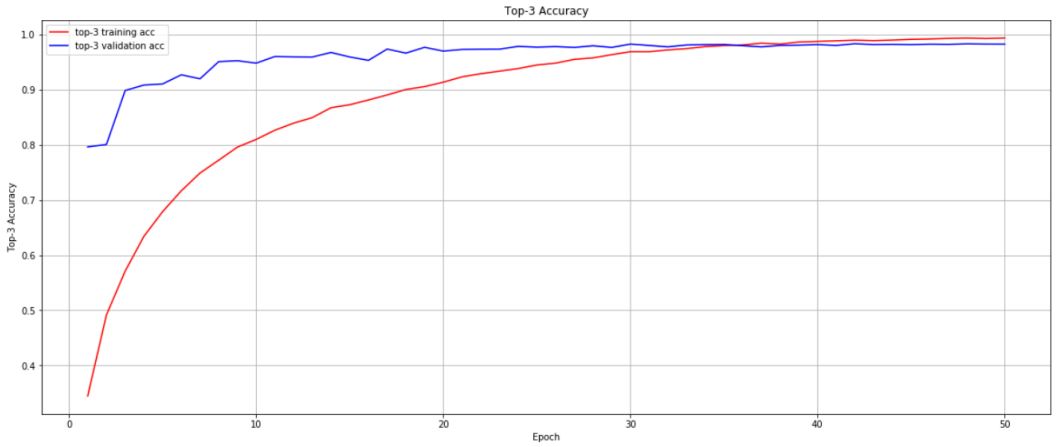


Figure 3: Top 3 accuracy

From my experiments, I found that adding dropout, activation functions, convolutional or fully connected layers, and adjusting optimizers have minimal impact on the final results. The most crucial aspect is data processing, including techniques like data augmentation and flipping. Figure 4 below presents the results of Vision Transformer on the CIFAR-10 dataset.

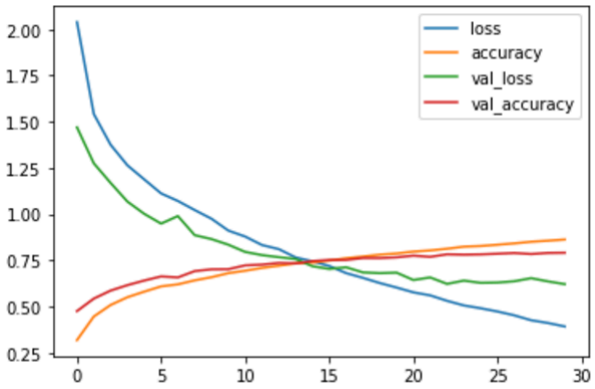


Figure 4: vision transformer

On the CIFAR-10 dataset, the Vision Transformer (ViT) shows some interesting results when compared with neural networks. According to search results, the performance of ViT on CIFAR-10 differs from that of Convolutional Neural Networks (CNNs). Traditional CNNs, due to their local receptive fields and weight sharing characteristics, typically perform well on small dataset tasks.

On the other hand, ViT processes images by dividing them into fixed-size patches and inputting these patches as serialized tokens into the Transformer model, utilizing self-attention mechanisms to handle the images. This approach excels at handling global information but may not be as efficient as CNNs on small datasets. A study proposed the addition of deep convolutional modules as shortcut connections in the ViT model to ensure that the model can capture both local and global information, significantly improving ViT's performance on small datasets like CIFAR-10.

Overall, on the CIFAR-10 dataset, ViT has advantages over neural networks in capturing global information, but on small datasets, it may require additional techniques such as data augmentation or structural improvements to enhance its performance. These results emphasize the importance of choosing the appropriate model architecture for different datasets and tasks.

After my findings, adjusting the parameters of the Vision Transformer (ViT) has a direct impact on the model's performance. Changing parameters such as the image segmentation method (Patch Size), the number of Transformer encoder layers, the number of heads in the multi-head attention mechanism, the hidden layer dimension (Hidden Size), and the learning rate (Learning Rate) all have a certain effect on the model's accuracy and efficiency. A larger Patch Size helps capture global information but may overlook details; conversely, a smaller Patch Size does the opposite. Increasing the number of encoder layers can enhance the model's expressive power but also increases computational complexity. Adding more heads can improve the model's ability to capture details but also increases computational costs. Increasing the hidden layer dimension can enhance the model's expressive power but similarly increases computational and memory overhead. Adjusting the learning rate affects the model's convergence speed and stability, which is similar to neural networks.

## Conclusion

And the validation of DenseNet on the CIFAR-10 dataset and CIFAR-100 dataset demonstrates the efficacy of this architecture in handling complex image classification tasks. The dense connectivity pattern, bottleneck layers, and transition layers collectively contribute to improved performance and faster convergence. This report underscores the importance of architectural design in deep learning models. The results highlight the benefits of DenseNet's architecture in terms of feature reuse and efficient learning. The dense connections between layers allowed the model to maintain information flow and gradient stability even when the network became very deep. Moreover, the use of bottleneck layers and transition layers contributed to reducing the computational burden while retaining the richness of the learned features. Subsequently, we will consider models of the other Transformer[3] model or large language model.

**References**

[1] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.

[2] Kingma D P. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[3]Vaswani A. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017.