Exploring Deep Learning Models for Efficient Image Classification on CIFAR-10 Dataset

1st Josip Ivancevic 2nd Nikolas Recke
Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro
Aveiro, Portugal
josip.ivancevic@fer.hr
nikolas.recke@rwth-aachen.de

***Abstract*—In this paper, we conduct a thorough comparison of three key deep learning models - Convolutional Neural Networks (CNN), Residual Networks (ResNet), and Dense Convolutional Networks (DenseNet) - for image classification tasks, using the widely recognized CIFAR-10 dataset [1]. We offer an in-depth analysis of existing research, detailing the pre-processing steps, model configurations, hyperparameter selection, and evaluation metrics used in our study. We also provide a comparative analysis of the performance of each model in terms of accuracy. Our results highlight varied performance across the models, indicating that while simpler models can achieve reasonable results, better performance generally demands more computational resources. Our study aims to deepen understanding and guide future research in the field of image classification algorithms.**

***Index Terms*—ML, deep learning, CNN, ResNet, DenseNet, CIFAR-10**

I. Introduction

A. Problem Statement

In the era of digital transformation, the volume of image data is increasing exponentially. This increase has boosted the development of automated systems for classifying images, with applications spanning from medical imaging to self-driving cars and security systems. However, the task of image classification with accuracy and efficiency remains tricky due to issues like high dimensionality, changes in lighting, pose, scale, and complex image backgrounds. In this paper, we approach the task of image classification using the CIFAR-10 dataset, a widely recognized benchmark dataset ideal for evaluating the performance of new models.

B. Scope of the Study

The scope of this study is to investigate the performance of three deep learning models - Convolutional Neural Networks (CNN), Residual Networks (ResNet), and Dense Convolutional Networks (DenseNet) - on the CIFAR-10 dataset. Each model brings different strengths to the table, from CNN's ability to effectively extract local features, ResNet's proficiency in mitigating the vanishing/exploding gradient problem, to DenseNet's feature reuse capability. Each model possesses unique advantages: CNN's strong local feature extraction, ResNet's solution to the gradient vanishing/exploding issue, and DenseNet's feature reuse ability. Our goal is to measure these models' performance in terms of accuracy and computational speed.

C. Motivation

Image classification holds a crucial role in many areas and is a technology of considerable value. It plays a key part in medical fields by aiding in early disease detection and supporting timely treatments. Moreover, it fuels the progress of self-driving vehicles, improves farming techniques, strengthens security measures, and supports environmental conservation efforts. Thus, image classification significantly contributes to sectors such as healthcare, transportation, agriculture, security, and environmental care, providing numerous societal benefits. Given the enormous future potential of image classification, we have chosen to work with a standard benchmark dataset, CIFAR-10. This allows us to become familiar with important algorithms for dealing with image classification challenges. The choice of this dataset is also driven by its data volume, which prepares us for handling large-scale datasets in the future, as well as provides a reliable basis for comparison and evaluation of our methods.

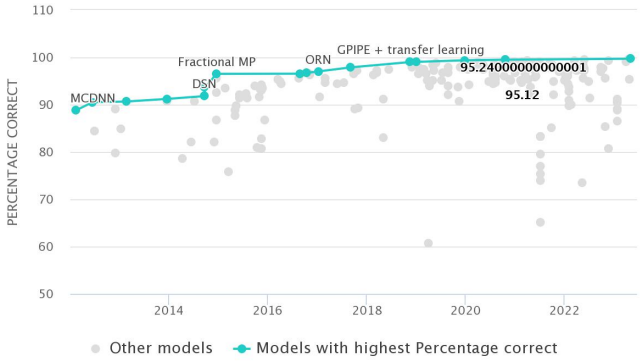

Fig. 1: Performance histories on the CIFAR-10 dataset. As can be observed, many researchers have tested their algorithms on this particular dataset. We only chose publications surpassing an accuracy threshold of 87.5%. [3]

II. State of the Art

Table I presents a comprehensive overview of the achievements of various models on the CIFAR-10 dataset. In our literature review, we aim to cover a wide range of models from different years to capture the advancements in image classification over time, such as CNN, ResNet, and DenseNet. To ensure a systematic comparison, the accuracies reported in the table were carefully selected based on a rising order, with a focus on including the best-performing models up to the latest available year. This approach allows us to showcase the progress made in model performance and identify the top-performing approaches.

A. Cifar-10 Classification using Deep Convolutional Neural Network

This paper focuses on image classification using convolutional neural networks (CNNs). The authors use the CIFAR-10 dataset to benchmark their deep learning model. They employ various function optimization methods such as Adam and RMSprop, along with regularization techniques, to achieve high accuracy in image classification.

The architecture of the CNN described in the paper consists of 6 convolutional layers, 3 pooling layers, and 3 fully connected layers. The Leaky ReLU activation function is used throughout the network, except for the output layer, where the Softmax activation function is utilized.

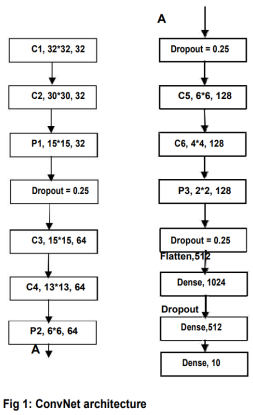

Fig. 2: Architecture of employed CNN [4].

The paper discusses several important features of the network architecture. These include Leaky ReLU Activation Function, Convolutional Layers, Batch Normalization, Max Pooling Layer, and Loss Function.

The paper also discusses techniques to reduce overfitting, such as regularization and dropout. Regularization involves adding a weight penalty term to the loss function to prevent overly complex models. Dropout randomly deactivates neurons during training to encourage independent feature learning and reduce reliance on other neurons.

Additionally, the paper mentions different function optimization methods, including gradient descent, stochastic gradient descent (SGD), and mini-batch gradient descent. SGD and mini-batch gradient descent are faster alternatives to gradient descent that use gradient approximations but require careful selection of the learning rate.

This paper provides a solid foundation for dealing with image classification tasks. The results achieved are worse in comparison to the other papers but the research paper provides an overview to start from.

B. Deep Residual Learning for Image Recognition

In "Deep Residual Learning for Image Recognition," He et al. (2015) introduced the ResNet architecture, which has significantly contributed to the development and evolution of deep learning, especially in the area of image recognition.

Most of the papers from that period advocated for deeper and wider networks for improved performance. However, they also highlighted the practical difficulties involved, such as the vanishing/exploding gradients problem and the added computational complexity. Addressing these issues, He et al. made the novel observation that extremely deep networks could struggle with training due to a degradation problem, where accuracy saturates and then degrades with increasing depth.

To solve this, they proposed the use of deep residual learning frameworks, where layers are reformulated to learn residual functions concerning the layer inputs, rather than the original unreferenced functions. This was implemented via shortcut connections (or skip connections), which carry the information from earlier layers to later ones, essentially allowing the network to skip layers during training. These residual networks, or ResNets, are easier to optimize and gain accuracy from increased depth. A residual learning block is shown in Figure 3.

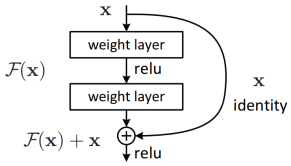

Fig. 3: Residual learning block for building residual framework [5].

One significant contribution of this work was the successful training of networks that are substantially deeper than those used previously, with network depths of up to 152 layers being realized. It's worth noting that the fundamental building block of these ultra-deep networks remains similar to those used in shallower networks.

For the CIFAR-10 dataset, He et al. demonstrated that a ResNet with 110 layers outperforms shallower architectures without residuals, achieving an impressive error rate of 6.43%. They adopted a more straightforward design without explicit bottlenecks for the CIFAR-10 experiments, unlike the design used for ImageNet.

The introduction of the ResNet architecture offered a paradigm shift in the design of deep networks, and its prin-

Paper	Dataset	Used Techniques	Accuracy on CIFAR-10
R. Doon, T. Kumar Rawat and S. Gautam, "Cifar-10 Classification using Deep Convolutional Neural Network" [4]	CIFAR-10	DCNN	87.57%
K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition [5]	CIFAR-10	ResNet	93.57%
G. Huang, Z. Liu, "Densely Connected Convolutional Networks [6]	CIFAR-10	DenseNet (k=40, 250 layers)	94.81%
S. Zagoruyko and N. Komodakis, "Wide Residual Networks [7]	CIFAR-10	WRN-16-8, WRN-28-10	95.81% and 96.00%
H. M. D. Kabir, "Reduction of Class Activation Uncertainty with Background Information" [8]	CIFAR-10	VIT-L/16	99.41± 0.12

TABLE I: Comparison of different model achievements on the CIFAR-10 dataset.

ciples have been widely adopted in various domains beyond image recognition.

C. Densely Connected Convolutional Networks

The paper "Densely Connected Convolutional Networks" by Huang et al. (2018) further advances the field of convolutional neural networks (CNNs). The authors note the advancements made in CNNs over the years: the movement from the inception modules in GoogLeNet, the introduction of residual connections in ResNet, and the increasing network widths in Wide ResNets. However, these previous approaches suffered from the problem of vanishing gradients and the need for careful design and arrangement of layers.

Huang et al. introduce the novel concept of DenseNet, a deeply connected CNN where each layer is connected to every other layer in a feed-forward fashion. The authors propose that the architecture reduces the vanishing gradient problem, strengthens feature propagation, and encouragesn terms of the novelty and contributions of our study, we did face some limitations that prevented us from making major breakthrough feature reuse. Unlike traditional CNNs, DenseNets do not rely on the summation of outputs from previous layers but concatenate them. This way, the network is capable of leveraging the learned features from all preceding layers.

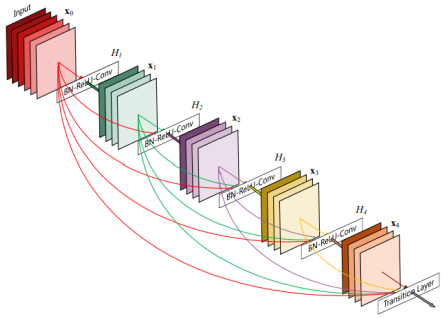

Fig. 4: Example of a Dense Block used in DenseNets.

In their architecture, each layer has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep supervision. Furthermore, this design significantly reduces the number of parameters, making the model less prone to overfitting, and also highly efficient for classification tasks, including CIFAR-10.

The paper exhibits extensive experimental evaluations, comparing DenseNet with existing state-of-the-art architectures. They achieved a significant reduction in error rates on CIFAR-10 and CIFAR-100 datasets. For the CIFAR-10 dataset, their best model achieved an error rate of 3.46

The authors also proposed a variant of DenseNet, called DenseNet-BC, which combines the concepts of bottleneck layers and transition layers with compression, which further enhances the model's efficiency.

D. Wide Residual Networks

In "Wide Residual Networks," Zagoruyko and Komodakis (2017) advance the field of deep learning architectures, specifically the concept of Residual Networks (ResNets) introduced by He et al. (2015). The authors build upon the works which emphasize the benefits of deeper networks for improving the performance of neural networks in image recognition tasks. However, contrary to the common belief that going deeper invariably results in better performance, Zagoruyko and Komodakis introduce the idea of making ResNets wider rather than deeper for improved performance. An intuitive understanding of their approach to widening ResNets can be done by seen in Figure 6

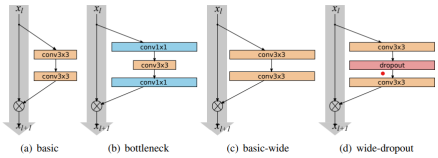

Fig. 5: Examples of blocks used in the paper. [7]

This novel approach was motivated by the observation that increasing depth could lead to various optimization problems, such as difficulty in training, vanishing/exploding gradients, and overfitting. The authors hypothesized that by increasing the width of residual networks – that is, by adding more convolutional filters in each layer – the network's capacity could be boosted without significantly adding to the aforementioned problems.

Their experiment results support this idea. They showed that wider residual networks outperformed deeper but thinner networks on various benchmark datasets, including CIFAR-10. They managed to achieve state-of-the-art performance on CIFAR-10 and CIFAR-100, with an error rate of 4.17% and 20.50%, respectively, using a wider variant of 16-layer-deep ResNets.

Another notable novelty introduced by the authors is the use of dropout in convolutional layers. By randomly dropping a

certain fraction of the activations (setting them to zero) during training, the authors found that the models' overfitting was reduced and their generalization capability improved.

E. Reduction of Class Activation Uncertainty with Background Information

H. M. Dipu Kabir's paper, "Reduction of Class Activation Uncertainty with Background Information," tackles a pivotal challenge in neural network training – the uncertainty of class activations. The novel solution proposed is the integration of a 'background class' into training, offering a more efficient approach than multitask learning while yielding superior generalization and at lower computational costs [8].

The primary innovation is the concept of a 'background class'. It addresses the issue of class activation uncertainty by providing the model with additional contextual information. In the image classification context, it assists the model in discerning the 'big picture' – aiding object identification by not just focusing on the object itself but also considering its surroundings. This background class is claimed to improve model generalization across various datasets, including CIFAR-10, CIFAR-100, STL-10, Oxford-102, Caltech-101, and CINIC-10. The application of this technique with traditional Convolutional Neural Networks (CNNs) and transformers led to state-of-the-art results [8].

Kabir also discusses the practicality of generating background classes. Currently, the process involves manual labor and checking whether an image contains the pattern of a target class or not. The author recommends future researchers to explore options for automating this process, perhaps by leveraging datasets where all objects in images are already labeled, such as the cityscapes dataset.

The paper achieved the best results out of all the other papers. The application of the ViT-L/16 model yielded an impressive accuracy of 99.41±0.12%. The introduction of background classes in the training process further elevated the model's accuracy, reaching an outstanding 99.67±0.07%.

One crucial issue raised is the difficulty in obtaining Class Activation Mappings (CAMs) from transformers. Class Activation Mappings (CAMs) are visualization techniques that highlight regions in an input image that contribute most to the model's output classification decision. As of the writing of this paper, no well-established method exists for achieving this. Therefore, statistical results of deep feature factorization could only be obtained from CNNs. Future research is urged to address this limitation and explore the impact of background classes on transformers [8].

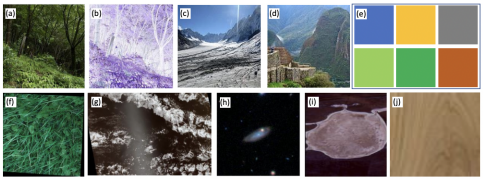

Fig. 6: Example images in the background class [8]

III. Data Description, Visualization, and Statistical Analysis

A. Overview of the CIFAR-10 Dataset

The CIFAR-10 (Canadian Institute for Advanced Research) dataset consists of 60,000 images of 32x32 pixels on 3 channels (RGB). These images are divided into ten classes, namely airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class contains 6,000 images. The dataset is split into 50,000 training images and 10,000 testing images. Each batch of data contains an assortment of images from each class, creating a well-distributed, diverse dataset that makes it a popular choice for image classification tasks. Figure 7 shows the total occurence of each picture in the dataset and highlights the balance of the dataset.

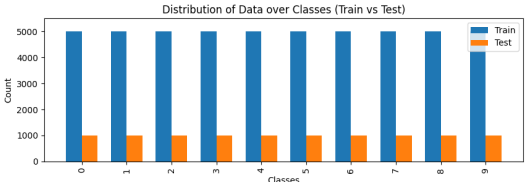

Fig. 7: Distribution of pictures .

B. Sample Data Presentation

Each image in the dataset is a 3-channel color image of 32x32 pixels in size. Each pixel is represented as a three-element array, where each element corresponds to the Red, Green, and Blue channels respectively. Each element is an integer between 0 (indicating a low intensity of color) and 255 (indicating a high intensity of color).

C. Metadata and Statistical Analysis

The metadata for the CIFAR-10 dataset is straightforward. Each image is associated with a single label from the ten possible classes.

The table below presents the computed mean and standard deviation values for each color channel in the CIFAR-10 dataset. The mean values indicate the average intensity of each channel, while the standard deviation values represent the measure of variation or spread within each channel.

TABLE II: Pixel intensities of the RGB-Channels of the CIFAR-10 dataset.

Channel	Mean	Standard Deviation
Red	0.491	0.247
Green	0.482	0.243
Blue	0.447	0.262

These values reflect the level of diversity or heterogeneity in the pixel intensities within each channel. The intensities of the Green channel are slightly higher than the intensities of the Red and Blue Channel and also scatter less, indicated by the smallest standard deviation.

D. Data Visualization

Visualizing the CIFAR-10 dataset is quite straightforward due to the relatively small size of the images. Sample images from each of the ten classes can be displayed to provide a sense of what each class represents. Furthermore, histograms showing the distribution of pixel values can be drawn to gain insights into the color composition of the images. These visualizations will provide valuable insights into the data and potentially influence preprocessing and modeling decisions. In Figure 8 we show a randomly selected image from each class of the dataset.

Fig. 8: Exemplary picture of each of the ten classes of the CIFAR-10 dataset.

After presenting exemplary images from the dataset to provide a visual context, we proceed to investigate the distribution of pixel intensities using histograms. This exploration allows us to gain valuable insights into the dataset and serves as a reasonable step in our analysis.

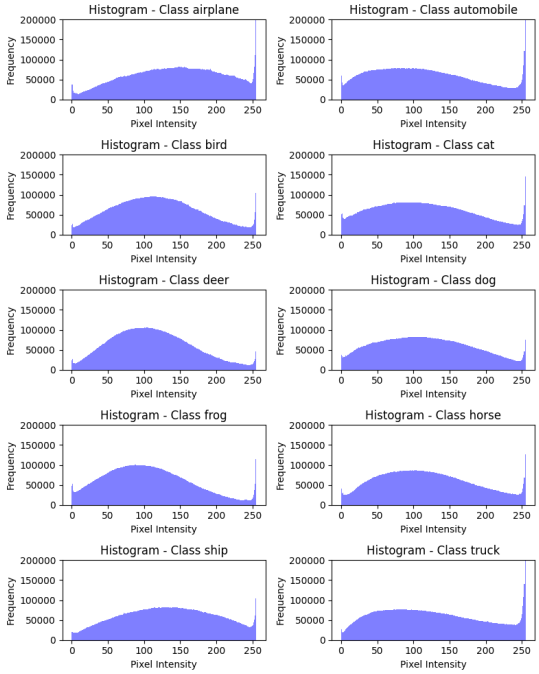

Fig. 9: Histograms of pixel intensities of each one of the ten classes of the CIFAR-10 dataset.

The histogram analysis reveals that there is no significant variation in the distribution of pixel intensities across classes. Therefore, we can conclude that there are no specific interesting subsets deserving further investigation or attention in this regard.

E. Data Quality

The CIFAR-10 dataset is a widely accepted benchmark for image classification due to its high-quality and precise data. It offers a comprehensive and reliable set of images with no missing data or corrupt images. Further, the dataset is well-labeled, ensuring there are no misclassifications or images that do not belong to the predefined classes.

Despite this, a study by Google Research [11] suggests that about 10% of the CIFAR-10 dataset could be seen as redundant. While this is a important observation, it hasn't been emphasized in many modern top-level studies using the CIFAR-10 dataset. Therefore, in line with these studies, we have decided to use the entire CIFAR-10 dataset, including the potentially redundant 10%, to maintain a fair comparison with other research in this field.

IV. Data Preprocessing

The preliminary step in any machine learning experiment is the data preprocessing phase, in which raw data is manipulated into a format suitable for the proposed machine learning model. This phase can encompass various techniques such as data normalization, feature selection, or dimensionality reduction, dependent on the nature of the data and the end goal.

The processes that we used in our study to preprocess data are as follows:

1) setting a deterministic behavior for the random number generators
2) preloading pretrained models for benchmarking
3) normalizing the data
4) data augmentation techniques to increase the diversity of the training set and prevent overfitting
5) splitting the data into training, validation, and testing sets
6) verifying the effectiveness of these steps through batch statistics and visual inspection

Since there are no significant redundant features in the CIFAR-10 dataset we decided not to proceed with the feature selection and dimensionality reduction for the scope of this paper.

A. Data Normalization

To execute the normalization step during the model's training and testing phases, we used the *transforms.Normalize* module from PyTorch's torchvision library, which normalizes the image's pixels given the mean and standard deviation.

B. Data Augmentation

After we normalized the data, we introduced data augmentation during the training phase. Data augmentation refers to the process of altering the training images to diversify the training data and prevent overfitting. We used two primary augmentations: transforms.RandomHorizontalFlip and transforms.RandomResizedCrop. The first one flips the images horizontally with a 50% chance, which increases the dataset's diversity without changing the object class. The second one adjusts the size of the image randomly and then crops it, which changes the pixel values but keeps the contents of the image the same.

It's worth noting that we chose to use data augmentation only for our more complex models, specifically ResNet and DenseNet, and not for the simpler CNN model. The reason behind this is that the CIFAR-10 dataset, which contains 50,000 images, offers sufficient data for a simple CNN model to learn effectively. As the complexity of models increases with additional layers, they require more data to learn effectively and perform well. As such, the deeper ResNet and DenseNet models stand to benefit from the additional data variety provided by the augmentation techniques, which assists them in learning more complex patterns and improving their performance.

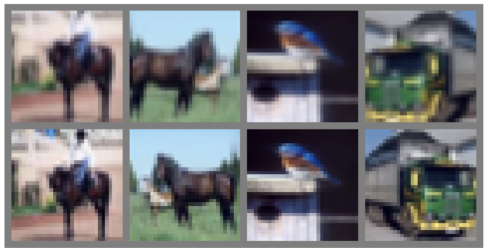

Fig. 10: Augmentation examples on CIFAR-10.

C. Train/Validation/Test Split

To make sure our model testing is thorough and fair, we divided our training dataset into two parts: a training set and a validation set. This division helps us to see how our model works on unseen data while it's being trained, and it lets us stop the training early if needed.

In the end, to check if our preprocessing worked, we calculated the mean and standard deviation of a single batch from the training set. The expected outcome is that the mean should approach zero, and the standard deviation should approach one for each RGB channel, confirming the normalization was successful.

TABLE III: Mean and standard deviation of a single batch from the training set.

Mean	[0.0231, 0.0006, 0.0005]
Std. dev.	[0.9865, 0.9849, 0.9868]

V. Description of the Applied Machine Learning Algorithms

A. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep learning models, primarily used for processing grid-like data, such as an image, which is a 2-dimensional grid of pixels. CNNs are designed to automatically and adaptively learn spatial hierarchies of features directly from images. This is done through the use of convolutional layers, where each layer is made up of a set of filters or kernels. These filters slide over the input data performing elementwise multiplication, and creating feature maps that give the network its ability to "see" aspects of the image. Figure 12 shows the structure of a convolutional neural network with several stages applied to the pixels and regions of a picture, including feature learning and classification.

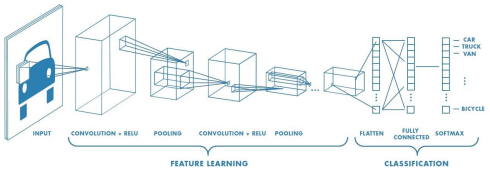

Fig. 11: Example structure of a CNN [10].

This structure allows CNNs to recognize local patterns in an image, such as edges, corners, and other textures, using fewer parameters than would be required with a fully connected network. Furthermore, because of parameter sharing, patterns can be recognized irrespective of their location in the image. CNNs are typically composed of three types of layers: convolutional, pooling, and fully connected layers.

For the scope of our project, we used a Sequential model from TensorFlow's Keras API, composed of three convolutional layers, each paired with a max-pooling layer. These layers learn spatial hierarchies from the images, a characteristic that makes CNNs particularly good for image processing.

The feature maps from the final convolutional layer are flattened to a one-dimensional array and passed through a fully connected (dense) layer. This layer has 64 units and uses the ReLU activation function, further processing the extracted features. The output layer, a dense layer of 10 units, provides the class probabilities for each of the 10 classes in the CIFAR-10 dataset.

Once the model structure is defined, we compile it using the Adam optimizer and sparse categorical cross-entropy loss function, with accuracy as the evaluation metric. The model is then trained on the dataset for 10 epochs, during which the loss and accuracy metrics are logged for both the training and validation sets. Lastly, the model's effectiveness is evaluated on the test dataset.

B. Residual Network (ResNet)

Residual Networks, or ResNets, are a type of CNN that introduced a novel architecture to mitigate the vanishing/exploding gradient problem faced by deep neural net-

works. As neural networks grow deeper, backpropagated gradients can vanish or explode, hindering the network's learning capability.

ResNets introduced the concept of "skip connections" or "shortcut connections," which allow the gradient to be directly backpropagated to earlier layers. These shortcut connections perform identity mapping, and their outputs are added to the outputs of the stacked layers. This architecture allows ResNets to train extremely deep networks (e.g., ResNet-152) and still achieve compelling performance by mitigating the issues faced by traditionally deep networks.

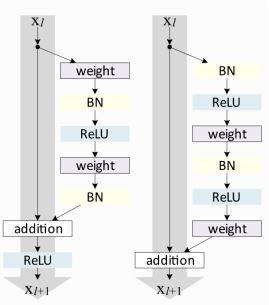

Fig. 12: Example structure of a ResNet [5].

Within the scope of this study, we utilized ResNet blocks which involve the application of a non-linear activation function—most commonly, ReLU—following the skip connection. The architectures of these blocks consist of 1x1 convolutions with a stride of 2, and they are arranged in three groups, each containing three ResNet blocks. These groups function on the resolutions of 32x32, 16x16, and 8x8, respectively. For the model training phase, we employed Stochastic Gradient Descent (SGD) complemented with Momentum as the optimization technique.

C. Dense Convolutional Network (DenseNet)

Dense Convolutional Networks (DenseNets) are another variant of CNNs that connect each layer to every other layer in a feed-forward manner. Unlike ResNets, which add outputs from previous layers, DenseNets concatenate the output of the previous layers. This architecture leads to what is referred to as "feature reuse," which means DenseNets require fewer parameters than traditional CNNs.

DenseNets can be quite efficient on image tasks, particularly when dealing with growth rates - the rate at which dimensions are added to the feature maps in deeper layers. This architecture allows for improved information flow between layers and encourages feature reuse, making DenseNets particularly well-suited for tasks that benefit from the preservation and use of fine-grained features throughout the network.

The implementation of DenseNet within our model is segmented into three essential constituents: a DenseLayer, a DenseBlock, and a TransitionLayer.

The DenseLayer forms the basic unit within a dense block. It introduces a 1x1 convolution for the task of dimensionality reduction, which is succeeded by a 3x3 convolution. The output channels from these operations are then concatenated with the original input channels, yielding the layer's final output.

A DenseBlock effectively encapsulates several dense layers operating sequentially. Every dense layer within the block accepts as its input the concatenation of the original input and the feature maps of all preceding layers within the block.

The role of the TransitionLayer is to process the final output of a dense block, specifically by diminishing its channel dimensionality via a 1x1 convolution.

A transition layer is appended after each dense block, excluding the last one, with the purpose of reducing dimensionality by a factor of 2. Upon completion of the network architecture, the Adam optimizer is utilized for the training phase.

D. Hyperparameter Optimization and Selection Process

After briefly explaining the models that we used in our paper, we proceed with the hyperparameter optimization. It is worth noting that this optimization process was employed only for the simple CNN model. For the more complex models, ResNet and DenseNet, we adopted data augmentation techniques to enhance model performance. These deeper models have more parameters and the capacity to learn, hence they require more data. By increasing the variety of data through augmentation, we allowed the models to learn more complex patterns and reduce overfitting. However, due to a lack of hardware capacity for performing cross-validation, we decided to omit the hyperparameter optimization for those models and choose the recommended hyperparameters from the literature.

In the process of training our Convolutional Neural Network (CNN), we performed hyperparameter optimization to identify the best combination of parameters that yield maximum validation accuracy. The set of parameters and the corresponding range we considered are listed in Table IV. These ranges were selected to cover a wide array of possible values, thus allowing the model to explore various configurations during the optimization process.

TABLE IV: CNN Hyperparameters Search Space

Parameter	Description	Range
conv1_units	Number of units in the first convolutional layer	[64, 128, 256]
conv2_units	Number of units in the second convolutional layer	[128, 256, 512]
conv3_units	Number of units in the third convolutional layer	[128, 256, 512]
learning_rate	Learning rate of the optimizer	[1e-2, 1e-3, 1e-4]
dense_units	Number of units in the dense layer	[64, 128, 256, 512]
dropout_rate	Dropout rate	[0.0, 0.5]

For this optimization, we employed the random search strategy. Instead of exhaustively searching through the entire hyperparameter space, random search picks a random combination of parameters for a certain number of iterations. This strategy is beneficial because it allows us to explore a broader variety of parameters and often finds a good set in less time than exhaustive methods.

Next, we utilized k-fold cross-validation to estimate the performance of the model for different hyperparameter configurations. We partitioned the data into 'k' equally sized 'folds' (we chose k=5 in this case). For each unique group, we took a fold as a validation set while the remaining folds form the training set. We iteratively validated the model on each fold and averaged the performance over the k iterations. This technique provides a robust estimate of the model's performance and reduces the risk of overfitting on a single validation set.

The CNN model was built and compiled in each iteration, using different combinations of hyperparameters each time. The parameters considered included the number of units in the convolutional and dense layers, the dropout rate, and the learning rate of the optimizer.

TABLE V: Optimal CNN Hyperparameters

Parameter	Optimal Value
conv1_units	128
conv2_units	512
conv3_units	128
learning_rate	0.001
dense_units	256
dropout_rate	0.4

The optimization process and the following best parameters are presented in Table V: a learning rate of 0.001, 128 units for both the first and third convolutional layer (Conv1 and Conv3), 512 units for the second convolutional layers (Conv2), a dropout rate of 0.4, and 256 units for the dense layer.

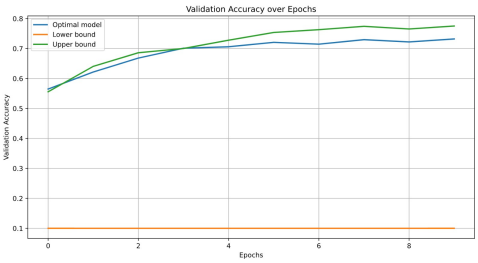

Fig. 13: Validation accuracy throughout the epochs in Random Search cross-validation

Figure 13 graphically depicts the progress in validation accuracy across epochs for three unique sets of hyperparameters: the optimally selected set, and two sets from the lower and upper limits of our search space. Interestingly, the set from the upper limit outperformed the model trained with the supposedly optimal parameters. This shows some limits of the random search approach we used.

However, the good accuracy results achieved by our models, even with their simple designs, suggest that our search space was properly defined. It also hints at the possibility of further improvements in hyperparameter tuning. With more computing power, we could expand the search space, use finer steps for tuning, and try more trials. These changes could likely improve the model's performance by identifying even better hyperparameters. This finding highlights the value of more extensive and resourceful parameter tuning in future work.

VI. Results and Discussion

After defining the dataset, preprocessing steps, and the models that we used in this paper, we finally present you with the results. We evaluated the performance of the following models in this paper:

- CNN
- CNN with optimal hyperparameters
- ResNet
- ResNet with an additional augmentation
- DenseNet
- DenseNet with an additional augmentation

A. Results of CNN

We started with a simple CNN model and predefined hyperparameters. After 10 epochs, this model achieved an accuracy of 70.49%. We then used Random Search cross-validation to find the optimal hyperparameters for the CNN. This optimized CNN model achieved an accuracy of 72.6%, showing a slight improvement over the basic CNN model. After this, we trained the ResNet model.

B. Results of ResNet and DenseNet on Augmented Dataset

In this section, we discuss the effect of augmentations on the dataset. We compared a basic augmentation strategy, consisting of a RandomHorizontalFlip and RandomResizedCrop, with a strategy that included several additional augmentations. The augmentations performed on the training data were:

- RandomCrop: Randomly crops the image with padding of size 4 pixels on each side to maintain the original size.
- RandomHorizontalFlip: Randomly flips the image horizontally with a probability of 0.5.
- RandomVerticalFlip: Randomly flips the image vertically with a probability of 0.5.
- RandomRotation: Randomly rotates the image by a maximum of 15 degrees.
- ColorJitter: Randomly adjusts brightness, contrast, saturation, and hue of the image with a maximum magnitude of 0.1.
- RandomAffine: Randomly applies affine transformations such as translation, scaling, and shearing to the image. The degree of transformation is controlled by the specified parameters.
- ToTensor: Converts the image to a tensor.
- Normalize: Normalizes the image tensor by subtracting the mean and dividing by the standard deviation calculated from the augmented dataset.

The slightly augmented dataset uses only two augmentations, while the augmented dataset uses all the augmentations listed above. Figures 15 and Figure 14 illustrate how the training data has been changed. It is important to mention that these enhancements can modify the properties of the pictures, causing the normalization to change. We've considered this and adjusted the normalization after enhancing the images.

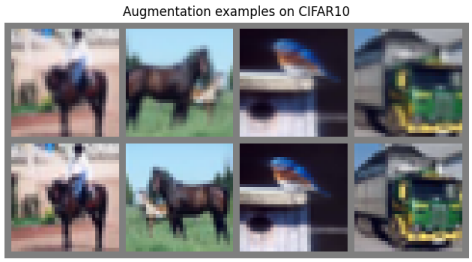

Fig. 14: Visualization of slightly Augmented Dataset.

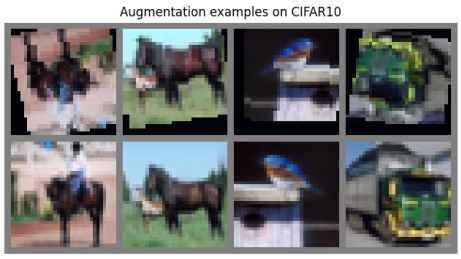

Fig. 15: Visualization of Augmented Dataset.

Table VII shows the accuracy results for different models and datasets. It lists the model type, dataset used, maximum number of epochs, and the resulting accuracy.

TABLE VI: Accuracy Results

Model	Dataset	Max Epochs	Accuracy
ResNet	Augmented	10	0.6986
ResNet	Augmented	20	0.7093
ResNet	Slightly Augmented	10	0.7664
DenseNet	Augmented	10	0.7266
DenseNet	Augmented	20	0.7763
DenseNet	Slightly Augmented	10	0.8043

Our tests were limited to 10 and 20 epochs due to GPU constraints on Google Colab. While we observed an improvement in the test accuracy of the augmented dataset with an increased number of epochs, the slightly adjusted dataset showed better performance with fewer epochs. This indicates that the chosen augmentations for the augmented dataset may have been too strong. Further evaluation of these augmentations will be beneficial and should be considered in future work.

C. Results of ResNet and DenseNet

Below, we present the accuracy outcomes obtained when using pretrained ResNet and DenseNet models on our test and validation datasets. These models were initially trained on a large dataset and then fine-tuned for our specific task. The next part showcases the training process on the CIFAR-10 dataset.

Figure 16 displays the training accuracy of ResNet (left) and DenseNet (right) across the number of training steps. The x-axis represents the steps taken during the training process, while the y-axis represents the corresponding train accuracy achieved by the models.

Figure 17 displays the validation accuracy of the ResNet (left) and DenseNet (right) over the number of training steps.

Fig. 16: Train Accuracies of ResNet (left) and DenseNet (right).

The x-axis represents the steps taken during the training process, while the y-axis represents the corresponding validation accuracy achieved by the models.

Fig. 17: Validation Accuracies of ResNet (left) and DenseNet (right).

The test accuracy shows how the models performed on unseen test data, and the validation accuracy displays their performance on the validation set during training. We based these results on the pre-trained models from [9]. The results are summarized in Table VII.

TABLE VII: Accuracy Results

Model	Max Epochs	Test Accuracy	Validation Accuracy
ResNet	180	0.9106	0.9184
DenseNet	180	0.9023	0.9072

The ResNet and DenseNet models showcased strong performance in the classification task. ResNet achieved a test accuracy of 0.9106 and a validation accuracy of 0.9184, while DenseNet attained a test accuracy of 0.9023 and a validation accuracy of 0.9072. So, ResNet slightly outperformed DenseNet. These results highlight the effectiveness of both models for image classification.

D. Comparative Analysis of the Performance of CNN, ResNet, and DenseNet

Table VIII provides a side-by-side comparison of the test accuracy for different models. The test accuracy measure shows how well each model can predict new, unseen data, giving us a good idea of how they might perform in real-world situations.

TABLE VIII: Comparison of Model Performance

Model	Test Accuracy
Simple CNN	0.705
CNN (Optimized hyperparameters)	0.726
ResNet	0.911
DenseNet	0.902

ResNet achieved the highest test accuracy of 0.911, closely followed by DenseNet at 0.902. These results highlight the effectiveness of pre-trained models like ResNet and DenseNet.

The Simple CNN and the CNN with optimized hyperparameters also performed well, with test accuracies of 0.705 and 0.726, respectively. Although not as high as ResNet and DenseNet, they can still be viable options in scenarios where computational resources or model complexity are constraints.

E. Comparison with the State-of-the-art

Lastly, we compare our results with those from the state-of-the-art in Figure 18. The blue dots show the results achieved by the papers analyzed in the state-of-the-art, while the stars represent the results from this project.

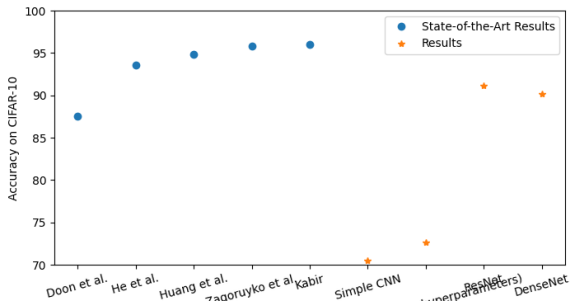

Fig. 18: Comparison with State-of-the-Art.

The results from the state-of-the-art models overall surpassed the generated results in this research project. However, with the pre-trained models, we were able to achieve similar results. When it comes to training a model from scratch, the computational demands exceeded our capabilities.

VII. Conclusions

A. Summary of the Study Findings

The study aimed to explore the effectiveness of deep learning models for image classification. We noted that while simple models can deliver satisfactory results, obtaining higher performance often necessitates considerable computational resources, such as advanced GPUs. The use of pre-trained models helped mitigate this constraint.

Pretrained models emerged as a valuable solution to address the GPU limitation. By utilizing models pre-trained on large-scale datasets, the study capitalized on the knowledge and features learned by these models, which resulted in improved performance on CIFAR-10 concerning older research results from the state of the art.

It is worth noting that hyperparameter optimization was relatively constrained due to the limited possibilities compared to smaller datasets, which may have influenced the overall performance.

In general, the findings underscored the power of deep learning as a tool for image classification tasks. The CIFAR-10 dataset proved to be a suitable benchmark for evaluating different methods and models. However, we also recognized that understanding more complex deep learning models required intensive study compared to simpler supervised learning algorithms.

B. Discussion of the Advantages and Disadvantages of the Applied Methods

As previously mentioned, a primary limitation in this study was the availability of GPU resources. This issue significantly influenced the training and optimization of the models. Run times exceeding several hours were impractical for our PCs. Although we utilized Google Colab, we frequently encountered the issue of unavailable GPUs, prompting us to use our PCs and endure longer runtimes.

CNN demonstrated certain advantages, including its simplicity in implementation, which led to reasonable results. Additionally, compared to more complex models, the CNN required a relatively short training time. However, it is important to note that CNN exhibited worse performance when compared to the other methods evaluated in the study. Furthermore, hyperparameter optimization for CNN still demanded a significant amount of time, which can be seen as a disadvantage.

Both ResNet and DenseNet showcased remarkable advantages. These methods achieved good results in terms of accuracy, outperforming the CNN, when used with a pre-trained model.

C. Novelty and Contributions

In terms of the novelty and contributions of our study, our study faced limitations that prevented us from making relevant breakthroughs, primarily the lack of access to extensive GPU resources. This limitation hindered explorations such as discovering an optimal hyperparameter combination. However, we made the most of the situation by working with a widely-used benchmark dataset that has been extensively studied before, to gain knowledge on the topic in general and extensively studying the methods and algorithms taught in class hands-on.

Our main goal was to contribute to the field by implementing and analyzing methods that we learned in class. We focused on techniques like data augmentation and hyperparameter optimization to improve the performance of our models. Through careful experimentation and analysis, we were able to demonstrate that proper tuning of hyperparameters can lead to better results in image classification tasks. Additionally, we emphasized the importance of being cautious when applying image augmentation techniques, as their effectiveness can vary depending on the specific dataset and task.

While our study may not have introduced groundbreaking concepts, our contributions rest in applying well-known methods and scrutinizing their effect on our chosen dataset. By building upon existing knowledge and systematically exploring different techniques, we believe we have provided valuable insights and practical considerations for future research in the field of image classification.

D. Recommendations for Future Research

Our study, conducted within the scope of a student project, yielded valuable insights into various deep learning models, particularly highlighting the superior performance of ViT-L/16 on the CIFAR-10 dataset. Nevertheless, our exploration was

constrained by limitations, primarily the lack of dedicated hardware resources, which prevented us from delving deeply into more computationally-intensive models or refining the training process for optimal performance.

Moving forward, overcoming these constraints could facilitate more thorough investigations. With access to specialized hardware, a more detailed examination of the top-performing model, ViT-L/16, would be feasible. This would enable us to determine the factors contributing to its exceptional performance on the CIFAR-10 dataset, and whether these attributes ensure equally strong results on similar datasets.

Moreover, investigating potential redundancy within the CIFAR-10 dataset, as suggested by Birodkar et al. [11], would be beneficial. Considering that up to 10% of the dataset might be redundant, there may be room for refining the training data to improve the efficiency of model training and the accuracy of performance metrics. Exploring this aspect could potentially yield better-performing models and more accurate results.

CIFAR-10 remains a benchmark dataset in the field, providing a solid basis for evaluating a wide range of models. As such, further work on this dataset is highly encouraged, as it could help in improving our understanding and performance of various models.

Work done by each student.

Nikolas Recke: Introduction, State-of-the-art (both), Data visualization, Model Descriptions, Results (both), Discussion and Conclusion (both), References (both), code (both)

Josip Ivancevic: State-of-the-art (both), Data Preprocessing, Model Descriptions, Model Evaluation Criteria, Hyperparameter Optimization, Results (both), Discussion and Conclusion (both), References (both), code (both)

REFERENCES

[1] Krizhevsky, A., Hinton, G., "Learning multiple layers of features from tiny images," Tech. Rep., University of Toronto, 2009. [Online]. Available: https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf.

[2] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[3] Papers with code - CIFAR-10 benchmark (image classification) (2023) The latest in Machine Learning. Available at: https://paperswithcode.com/sota/image-classification-on-cifar-10 (Accessed: 06 June 2023).

[4] R. Doon, T. Kumar Rawat and S. Gautam, "Cifar-10 Classification using Deep Convolutional Neural Network," 2018 IEEE Punecon, Pune, India, 2018, pp. 1-5, doi: 10.1109/PUNECON.2018.8745428.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778.

[6] G. Huang, Z. Liu, and L. van der Maaten, "Densely Connected Convolutional Networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4700-4708.

[7] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," in Proceedings of the British Machine Vision Conference (BMVC), 2016, pp. 87.1-87.12.

[8] H. M. D. Kabir, "Reduction of Class Activation Uncertainty with Background Information," arXiv preprint arXiv:2305.03238v2, 2023.

[9] P. Lippe, "UvA Deep Learning Tutorials," 2022. Available at: https://uvadlc-notebooks.readthedocs.io/en/latest/ (Accessed: 06 June 2023).

[10] S. Saha, "A comprehensive guide to Convolutional Neural Networks - the eli5 way," Saturn Cloud Blog, https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/ (accessed Jun. 7, 2023).

[11] V. Birodkar, H. Mobahi, and S. Bengio, "Semantic Redundancies in Image-Classification Datasets: The 10% You Don't Need," arXiv preprint arXiv:1901.11409, 2019.