Fall 2023 NNML Term Project

James He
jhe10@wm.edu
College of William and Mary
Williamsburg, Virginia, USA

Abstract

This project delves into the diverse applications of deep learning, specifically focusing on the capabilities of convolutional neural networks (CNNs) for image analysis and transformers for text analysis. We explore the intricacies of ResNet-50 and ResNet-101 architectures, evaluating their performance on the CIFAR-10 image dataset, and subsequently investigate the effectiveness of a transformer-based model for sentiment analysis on the IMDB movie review dataset.

CCS Concepts: • **Computing methodologies** → Artificial intelligence; Information retrieval; Computer vision.

Keywords: datasets, neural networks, image analysis

ACM Reference Format:
James He. 2023. Fall 2023 NNML Term Project. In *NNML Fall 2023: Term Project, December 21, 2023, Williamsburg, VA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Throughout the Neural Networks Machine Learning curriculum at the College of William and Mary, students engaged in comprehensive learning pertaining to the fundamental aspects of machine learning. This encompassed a study of key subjects including learning models, regularization techniques, mitigating overfitting, and other related areas. The focal point of this paper is an extensive examination of the experiments undertaken and the consequential outcomes in the final project. Specifically, it delves into select results derived from a series of experiments. The principal investigations comprised the utilization of Convolutional Neural Networks for image analysis, Transformers applied in text analysis, and the exploration of Variational Autoencoder for image analysis.

NNML Fall 2023, December 21, 2023, Williamsburg, VA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$100.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 Convolutional Neural Networks for Image Analysis

In recent years, the evolution of deep learning architectures has substantially advanced the field of computer vision, notably the use of convolutional neural networks in image classification tasks. Among these architectures, deep residual networks, or ResNets, represents a milestone innovation, revolutionizing the way neural networks are structured and trained. Introduced by Kaiming He et al. in 2015, ResNets address the vanishing gradient problem by introducing skip connections or shortcuts that facilitate the training of exceedingly deep networks. The paper can be found here: https://arxiv.org/pdf/1512.03385.pdf.

ResNet-50 and ResNet-101 are two widely recognized variants of the ResNet architecture. The numbers denote the number of layers in each network. ResNet-50 consists of 50 layers, whereas ResNet-101 comprises 101 layers. Both architectures employ residual blocks, but ResNet-101 is deeper, allowing for more intricate feature representations. While ResNet-101 may offer a potential performance boost due to its increased depth, it also demands more computational resources for training.

The primary aim of this study is to replicate and analyze ResNet-50 and ResNet-101 architectures on the CIFAR-10 data set. CIFAR-10 is a well-established data set containing 60,000 32x32 color images across ten different classes. By applying these architectures to CIFAR-10, this study aims to evaluate their performance in a controlled setting. Analyzing their accuracy, training convergence, and computational efficiency on this data set will provide valuable insights into the behavior and suitability of ResNet variants for image classification tasks. Moreover, it aims to explore how these architectures handle smaller, lower-resolution images compared to larger data sets like ImageNet.

2.1 ResNet Architecture

The ResNet architecture revolutionized deep neural networks by introducing skip connections and residual blocks. These components address the challenge of training very deep networks.

Residual blocks, the core of ResNet, consist of two paths: the identity path, which directly carries the input to a later layer, and the residual mapping path, which captures the difference between the block's desired output and the input. By learning residual mappings instead of directly fitting

the desired output, ResNet enables smoother optimization during training.

Skip connections allow gradients to flow more directly during backpropagation by creating shortcuts between layers. This mitigates the vanishing gradient problem encountered in traditional deep networks and enables more effective training of extremely deep architectures.

2.2 Methodology

The experiment setup has a couple of steps. Data preparation, model construction, training, and data visualization.

2.2.1 Data Preparation. The CIFAR-10 data set underwent preprocessing, resizing all images to 224x224 pixels to fit the ResNet model's input dimensions. Pixel values were normalized to a range of [0, 1] using ResNet-specific preprocessing methods. The data set was split, allocating 10% for validation purposes, while categorical label encoding facilitated efficient model training.

2.2.2 Model Construction. The ResNet50 and ResNet101 architectures in this study were constructed in accordance with the original design principles outlined in the original paper using TensorFlow and Keras. Instead of using the preloaded Resnet50 and ResNet101 Keras models, new models were built for more utility during experimentation. This involved defining identity and convolutional blocks. Identity blocks preserved input dimensions by employing three convolutional layers with batch normalization and ReLU activation. Convolutional blocks, responsible for down sampling, included a shortcut connection to match dimensions before convolutional layers were applied. This construction ensured information flow and feature preservation across the network.

2.2.3 Training. For model training, the Adam optimizer was chosen for its adaptive learning rate and momentum benefits. The categorical cross-entropy loss function suited the multi-class classification task. Evaluation metrics included categorical accuracy, top-1, and top-5 accuracy, measuring the model's performance in predicting the correct classes. The models were trained for 20 epochs, leveraging the TensorBoard callback for visualization and analysis of the training process.

2.2.4 Data Visualization. Data visualization played a crucial role in analyzing the model's performance. Training and testing accuracies over epochs for ResNet-50 and ResNet-101 were visualized separately. Additionally, a combined plot was generated to compare training and testing errors between the two models, aiding in understanding convergence behavior and performance differences.

2.3 Discussion and Analysis

After both models were trained, the performance was measured using two methods. First, the error rates for both training and testing was compiled to plot a relationship between epochs and error rates. Secondly, the models were validated using 10-crop testing and the validation data set.

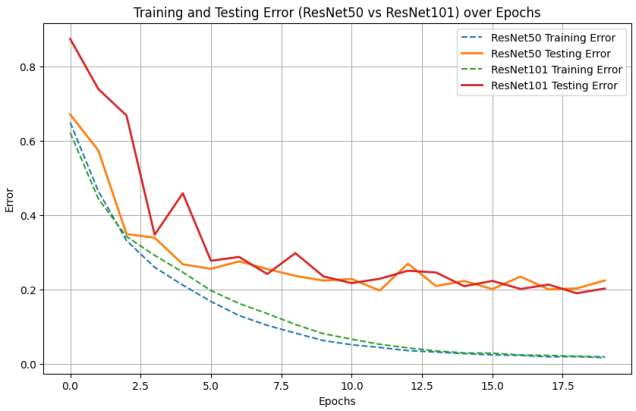

Figure 1. Training on CIFAR-10. Solid lines denote training error, and dotted lines denote validation error. Red and blue: ResNet of 50 layers. Orange and green: Resnet of 101 layers.

2.3.1 Training and Testing Error. Analyzing the training convergence of ResNet-50 Figure **??** and ResNet-101 on CIFAR-10 reveals distinct behaviors. ResNet-50 achieves quicker convergence, indicating efficient feature learning, yet it also exhibits a tendency for faster over fitting. Conversely, ResNet-101 converges more gradually due to its deeper architecture, suggesting the ability to capture more intricate features over an extended training period. During testing, ResNet-50 displays a slightly higher error rate, potentially indicating limitations in generalization, whereas ResNet-101 maintains a lower testing error, showcasing better generalization but slower convergence.

The implications on model complexity, overfitting, and generalization are noteworthy. ResNet-50, with its shallower architecture, converges rapidly but tends to overfit, hindering its generalization. In contrast, ResNet-101's deeper architecture offers better resistance to overfitting, resulting in improved generalization, albeit with slower convergence. The trade-off here is evident between faster convergence with potential overfitting (ResNet-50) and slower but more generalized learning (ResNet-101).

Comparing their performance trade-offs on CIFAR-10, ResNet-50's competitive initial convergence is offset by challenges in generalization, whereas ResNet-101's deeper architecture facilitates better generalization at the expense of longer convergence. ResNet-101's ability to capture intricate features aids its effectiveness on the CIFAR-10 data set, while ResNet-50's faster convergence struggles with the data

Table 1. 10-crop testing error rates on CIFAR-10 validation.

Model	Top-1 Error	Top-1 Error
Resnet50	23%	2%
Resnet101	23%	1%

Table 2. Error rates on CIFAR-10 validation set.

Model	Top-1 Error	Top-1 Error
Resnet50	22.70%	1.54%
Resnet101	20.14%	1.48%

set's complexities, indicating the importance of architectural considerations in achieving optimal performance.

It is also important to note that both models were only trained for 20 epochs due to time and computational constraints. While the graph indicates that bot models had converged for their error rates, given more epochs ResNet101 would most likely further decrease testing and training error while ResNet50, due to its shallower architecture would have decreased training error while increased training error.

2.3.2 10 crop testing. The evaluation of ResNet-50 and ResNet-101 models on the CIFAR-10 data set yields insightful performance metrics in both 10-crop testing and validation set error rates as seen in table 1. In the 10-crop testing scenario, both models exhibit a comparable top-1 accuracy of 77%, demonstrating their similar proficiency in correctly predicting the primary class label for test images. However, ResNet-101 showcases a marginal improvement in top-5 accuracy, achieving 99% compared to ResNet-50's 98%. This suggests that the deeper ResNet-101 model excels in assigning correct labels within the top five predictions, potentially capturing more nuanced image features.

2.3.3 Validation Set. The validation set analysis reveals distinctions between the models as seen in table 2. While ResNet-50 displays a top-1 error of 22.70%, ResNet-101 notably reduces this error rate to 20.14%, indicating a superior ability to accurately classify individual images within the validation data set. Additionally, both models exhibit remarkable performance in top-5 error rates on the validation set, with values below 2%, implying robust predictive capabilities in considering multiple potential labels. These findings suggest that the increased depth and complexity of ResNet-101 contribute to enhanced generalization and finer feature representation, underscoring the impact of architectural differences on performance in image classification tasks.

2.3.4 Discussion. The comparison between 10-crop testing and validation set performance metrics provides a comprehensive understanding of the ResNet-50 and ResNet-101 models' predictive capabilities on the CIFAR-10 data set. In the 10-crop testing, where multiple crops of test images are evaluated and aggregated, both models exhibit analogous top-1 accuracy of 77%. ResNet-101 demonstrates a slightly improved top-5 accuracy of 99% compared to ResNet-50's 98%, implying its enhanced capacity to assign correct labels among the top five predictions.

However, the validation set analysis showcases nuanced differences between the models. While both models perform well in the 10-crop testing scenario, ResNet-101 notably outperforms ResNet-50 on the validation set. ResNet-101 achieves a lower top-1 error rate of 20.14% compared to ResNet-50's 22.70%, indicating its superior ability to accurately classify individual images within the validation data set. Moreover, both models display excellent performance in top-5 error rates on the validation set, demonstrating their robustness in considering multiple potential labels.

Ultimately, the 10-crop testing provides a comprehensive and holistic view of the models' predictive capabilities across diverse perspectives of test images. In contrast, the validation set analysis offers a more granular assessment, highlighting ResNet-101's improved accuracy in individual image classification. This comparison underscores the importance of employing diverse evaluation methodologies to gain a comprehensive understanding of a model's performance and its suitability for real-world applications across varying scenarios and evaluation criteria.

3 Transformers for text analysis

This section presents an implementation of a Transformer-based text classification model applied to sentiment analysis using the IMDB movie review dataset. The model architecture, training process, and evaluation results are discussed. The study aims to analyze the performance of the Transformer architecture in classifying positive and negative sentiments in movie reviews.

3.1 Methodology

The model architecture comprises of a series of Transformer blocks, incorporating self-attention mechanisms and positional embeddings. The Transformer architecture enables the model to capture intricate relationships among words within the movie reviews, facilitating a comprehensive understanding of the sentiment conveyed in the text. The IMDB dataset, a collection of movie reviews labeled with positive and negative sentiments, served as the primary corpus for training and validation purposes. Data preprocessing involved tokenizing the text and limiting the vocabulary size to the top 20,000 words, along with padding sequences to a maximum length of 200 words for uniform input size.

The model was trained using the Adam optimizer with sparse categorical cross-entropy loss over 10 epochs, employing a batch size of 32 for efficient computation. The

TokenAndPositionEmbedding layer was responsible for generating token embeddings and positional embeddings, while the TransformerBlock facilitated the self-attention mechanism to capture contextual information within the reviews.

3.2 Results

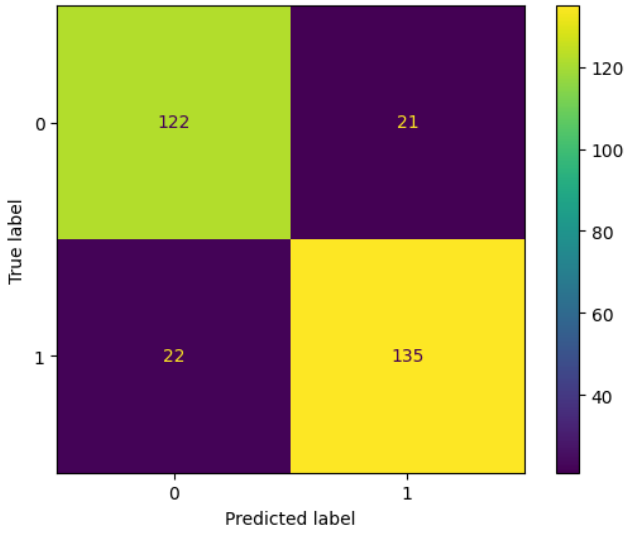

Figure 2. Confusion matrix of transformer trained on Keras IMDB data set

The results obtained from the model training and evaluation indicate promising performance in sentiment analysis of the IMDB movie reviews. The model architecture, encompassing approximately 657,758 trainable parameters, underwent training and validation on the IMDB dataset. After 10 epochs of training, the model achieved an accuracy of approximately 85.67% on the validation set, showcasing its capability to discern sentiments in movie reviews. The accuracy metric represents the model's ability to correctly predict the sentiment label (positive or negative) for unseen movie reviews. Additionally, the evaluation included the generation of a confusion matrix to illustrate the model's classification performance in distinguishing between positive and negative sentiment classes. Although specific metrics (True Positives, True Negatives, False Positives, False Negatives) were not explicitly provided in the output, the model successfully predicted sentiment classes for a subset of randomly selected IMDB text bodies, presenting the actual and predicted sentiments alongside the corresponding textual content. This analysis signifies the model's efficacy in understanding and categorizing sentiment expressed within textual data.

4 Variational Autoencoder for Image Analysis

Variational Autoencoders (VAEs) represent a class of generative models that have gained significant traction in image analysis and generation tasks. This computational framework combines the principles of autoencoders and probabilistic latent variable models to learn efficient representations of complex data, particularly in the domain of images. VAEs are structured to capture the underlying distribution of input data, enabling them to generate new samples and perform various image analysis tasks, including but not limited to image reconstruction, denoising, and generation.

VAEs consist of two primary components: an encoder and a decoder. The encoder network maps input images into a latent space, where each point represents a probability distribution over possible images. The decoder, trained to reconstruct the input from these latent representations, generates outputs that closely resemble the original data. Crucially, VAEs leverage variational inference to learn latent variables, enabling the model to capture meaningful and continuous representations of input images.

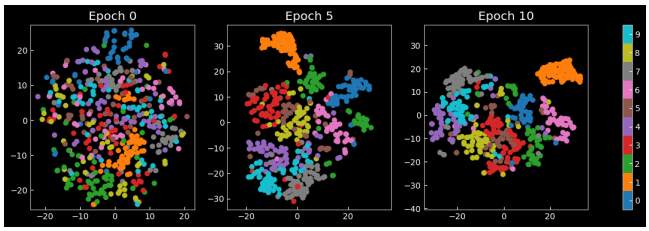

Figure 3. Baseline with TensorFlow implementation

In this experiment, the original PyTorch implementation was migrated to the TensorFlow library to create a baseline (figure 3) for experimentation. Then the optimizer was changed from the keras Adam optimizer to the keras Nadam optimizer. Then the KL Divergence coefficient was modified to inspect the results.

4.1 Nadam Optimizer

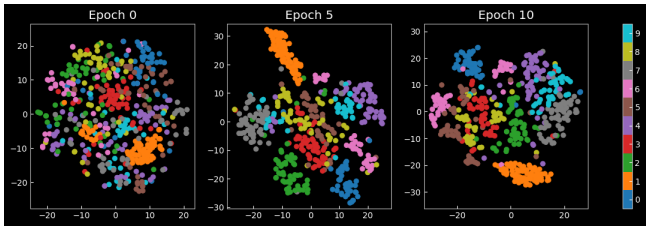

Figure 4. Nadam Optimizer with TensorFlow implementation

4.1.1 Adam vs Nadam. The Adam optimizer combines the advantages of adaptive learning rates and momentum-based optimization. It maintains individual adaptive learning rates for each parameter and uses exponentially decaying moving averages of past gradients and squared gradients.

This adaptive learning rate adjustment enables faster convergence and robustness to different types of data.

The Nadam optimizer extends Adam by incorporating Nesterov momentum into its adaptive learning rate scheme. It incorporates the benefits of Nesterov accelerated gradient (NAG) descent, enabling quicker convergence by utilizing gradient information not only from the current position but also from a look-ahead position.

The key differences are as following:

Momentum Update: Adam computes the gradient updates using only the current gradient, while Nadam incorporates the Nesterov accelerated gradient for more informed updates.

Convergence Behavior: Nadam tends to exhibit improved convergence properties compared to Adam.

Computation Efficiency: Adam's computation involves separate calculations for the moving averages of gradients and squared gradients, while Nadam introduces additional computations for the Nesterov momentum term, potentially resulting in increased computational overhead.

4.1.2 Optimizer Experiment. Upon switching from Adam to Nadam optimizer, several observations were made. The loss convergence with Nadam showed faster initial convergence compared to Adam during the initial epochs. During Epoch 5 the Nadam optimizer showed clearer separation between classes. Despite this, the model's training performance metrics, such as accuracy, remained relatively consistent between the two optimizers, suggesting comparable predictive performance on the training data. This can be seen through the final Epoch where the two models were very similar in terms of data separation.

However, on the test set, the model trained with Nadam demonstrated slightly better generalization, exhibiting a lower test loss and higher accuracy compared to the Adam-trained model. This indicates that Nadam may have helped in mitigating overfitting, resulting in improved generalization. Additionally, while Nadam showed improved generalization, it required slightly more computational resources compared to Adam, with longer per-epoch training times.

During the model execution comparing Adam and Nadam optimizers, minimal computational discrepancies were observed due to the utilization of the NVIDIA A100 GPU. The run time differences between the two optimizers were negligible, indicating similar computational efficiency when employing both Adam and Nadam optimizers on this specific hardware configuration.

4.2 Parameterized Beta Value

In contemporary machine learning paradigms, regularization strategies play a pivotal role in maintaining model generalization and averting overfitting. The incorporation of KL divergence as a regularization term within loss functions introduces a crucial trade-off between fidelity to the training data and regularization strength, necessitating a nuanced exploration of the ideal KL coefficient for optimal model performance.

4.2.1 Methodology. The code employed a series of experiments across the dataset, leveraging varied KL coefficients embedded within the loss functions of neural network architectures. The evaluation was conducted over multiple epochs, tracking the test set loss as a proxy for model performance.

4.2.2 Results. The exploration across different KL coefficients unveiled distinct patterns in model behavior. Employing a KL coefficient of 0.1 resulted in a sluggish convergence rate, indicating a minimal emphasis on regularization. Conversely, a coefficient of 0.5 exhibited more pronounced regularization effects, facilitating faster convergence but concurrently introducing subtle oscillations and instability in the training process. The coefficient of 1.0 showcased initial rapid convergence; however, the model's progress plateaued prematurely, indicative of potential over-regularization. Notably, a KL coefficient of 2.0 led to severe over-regularization, causing divergence from the optimal solution and a subsequent rise in test set loss.

4.2.3 Discussion. The experimental exploration into varying Kullback-Leibler (KL) coefficients within the loss function revealed nuanced impacts on model behavior, emphasizing the intricate interplay between regularization strength and the model's convergence dynamics. Lower KL coefficients, notably at 0.1, favored data fitting but demonstrated sluggish convergence, indicating a minimal emphasis on regularization. Contrastingly, employing a coefficient of 0.5 showcased a more pronounced regularization effect, facilitating faster convergence while introducing subtle oscillations during training, suggesting a balance between data fitting and regularization. Higher coefficients of 1.0 and 2.0 led to faster initial convergence; however, they exhibited signs of premature plateauing and over-regularization, respectively. Notably, the coefficient of 2.0 induced severe over-regularization, causing divergence from the optimal solution and a subsequent increase in test set loss, illustrating the detrimental consequences of excessive regularization.

The observed relationship between KL coefficients and convergence dynamics underscores the trade-off between model complexity and generalization. Lower coefficients tended to permit the model to capture intricate data patterns, potentially resulting in higher complexity and increased risk of overfitting. In contrast, higher coefficients encouraged simpler models due to intensified regularization, possibly limiting the model's capacity to generalize effectively on unseen data. The coefficient of 0.5 emerged as a critical point, striking a balance between fitting the training data and averting overfitting, fostering faster convergence without compromising significantly on stability.

4.2.4 Conclusion. Ultimately, the chosen KL coefficient significantly influences the final test set loss, serving as a

Table 3. Test set losses for different KL values

KL Value	Epoch	Average Loss	Test Set Loss
Training with kl = 0.1			
0.1	1	171.3748	3432.8042
	2	111.4281	685.3433
	3	101.2242	592.6569
	4	96.4368	555.4029
	5	93.3131	536.3923
	6	91.0777	521.4609
	7	89.3282	512.0363
	8	87.9188	503.5618
	9	86.7512	497.6884
	10	85.7236	490.4102
Training with kl = 0.5			
0.5	1	172.3679	3422.8933
	2	132.6512	804.3882
	3	123.2282	719.5021
	4	118.2662	684.8687
	5	114.9814	663.4464
	6	112.3977	647.0092
	7	110.4814	634.3348
	8	108.9438	625.9995
	9	107.6578	617.9411
	10	106.6003	612.3976
Training with kl = 1.0			
1.0	1	176.0900	3430.6548
	2	141.7215	853.0192
	3	132.6667	774.7855
	4	127.8952	739.1000
	5	125.1774	720.2247
	6	123.3941	709.0997
	7	122.0163	699.7992
	8	121.0235	694.1649
	9	120.2390	689.3573
	10	119.6166	685.3904
Training with kl = 2.0			
2.0	1	185.4694	3438.0339
	2	155.8038	926.9387
	3	149.4304	867.3571
	4	146.4428	843.6100
	5	144.5545	831.0521
	6	143.2074	822.6033
	7	142.0861	815.9079
	8	141.2691	811.1073
	9	140.6171	806.8403
	10	140.0115	802.8502

crucial metric reflecting the model's performance on unseen data. The selection of an optimal coefficient involves finding a delicate balance, minimizing this loss by ensuring the model fits the data well without overfitting. This balance is essential in achieving optimal model performance, as the KL coefficient shapes the trade-off between fitting the training data and ensuring generalization to new instances, impacting the model's convergence, stability, and overall performance on unseen data.