CNN and Random Feature Models for CIFAR-10 Classification

Jacob Goulet, Tyler Rossi, Diego Bueno, Javier Pozo Miranda, Duong Bao, Garrett Fincke

December 19, 2024

Abstract

This final project compares two individual approaches to image classification using the CIFAR-10 dataset. The first aim is to construct a Convolutional Neural Network inspired by ResNet and DenseNet, using state-of-the-art architectures and techniques such as residual connections, dense connections, dropout regularization, and data augmentation to achieve high classification performance. The second project looks at the application of the Random Feature Model (RFM), a simplified kernel-based counterpart using Random Fourier Features to approximate the Radial Basis Function (RBF) kernel. The aims of both projects are to compare the trade-offs between deep learning models and kernel approximation methods with regards to accuracy, efficiency, and computational complexity, hence providing an understanding of their applicability for image classification. The results demonstrated that DenseNet achieved the highest accuracy of 74%, while the RFM offered computational simplicity but a lower accuracy of 51.57%.

Keywords: CIFAR-10, ResNet, Convolutional Neural Networks, Random Feature Models, Deep Learning, Image Classification

1 Introduction

The CIFAR-10 dataset is one of the most popular datasets for benchmarking image classification models; it consists of 60,000 32x32 color images falling into 10 classes. This project investigates two different approaches: a ResNet-inspired CNN and a Random Feature Model. The first project will investigate using a ResNet model with some advanced techniques to get high performance. The second project details will be updated later.

2 Related Work

Convolutional Neural Networks (CNNs) have achieved spectacular results on the task of image classification, with architectures such as AlexNet, VGG, ResNet, and DenseNet all pushing state-of-the-art performance in multiple benchmarks. Introduced in ResNet, residual connections came into being as the means to offset the vanishing gradient problem, hence training much deeper networks thanks to very effective gradient propagation.

On the other hand, DenseNet uses dense connections to improve feature re-usage and gradient flow for more compact and robust feature learning. These changes have made CNNs one of the most successful strategies for image classification tasks, especially on benchmarks such as CIFAR-10.

RFMs—for instance, using Random Fourier Features—are an efficient approach for approximating nonlinear kernel functions on high-dimensional data. Being computationally lightweight, RFMs depend on the use of random transformations of the data to map into a feature space that lets a linear model approximate sophisticated kernel-based solutions. They do not have the same order of computational demand as, for example, deep learning models, and this shows an even more desirable path when resources are an issue in practical implementation scenarios. Unlike CNNs, which require huge computational resources in training and inference, RFMs require very few parameter tunings and have superior scalability. This makes RFMs extremely suitable for environments where either hardware or time constraints prohibit the application of deep learning methods.

Two approaches are discussed for CIFAR-10 classification: ResNet-inspired CNNs and DenseNet as representatives of state-of-the-art deep learning architectures, and RFMs as a lightweight, kernel-based alternative. CNNs are extremely good at hierarchical feature extraction and, hence, better suited for capturing the spatial dependencies inherent in image data. RFMs, on the other hand, achieve a good trade-off between simplicity and performance, hence a feasible solution for the classification task where computational efficiency is a concern. Such comparison of the approaches clearly brings out the trade-offs between model complexity, computational requirements, and effectiveness—hence, practically bringing out the strengths and weaknesses of each approach. limitations in practice.

3 Problem Set Up and Data

The project involves two distinct tasks: (1) classifying images from the CIFAR-10 dataset using CNNs and RFMs and (2) analyzing ECG signals using RNN/LSTM models for time-series classification.

3.1 Task 1: CIFAR-10 Classification

The CIFAR-10 dataset contains 60,000 32x32 color images divided into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is split into 50,000 training images and 10,000 test images.

Preprocessing steps include:

- Normalizing pixel values to the range [0, 1].
- Flattening images for RFM input while retaining the spatial structure for CNNs.
- Splitting the training data into training and validation subsets for RFM loss tracking.

The problem setting for each model:

- **ResNet-inspired CNN and DenseNet**: Deep networks with advanced architectures (e.g., residual and dense connections) to extract hierarchical features and improve performance.
- **RFM**: A lightweight model using Random Fourier Features to approximate the RBF kernel for efficient kernel-based classification.

3.2 Task 2: Random Feature Model for CIFAR-10 Classification

The CIFAR-10 dataset contains 60,000 32x32 color images divided into 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is split into 50,000 training images and 10,000 test images.

Preprocessing steps include:

- Normalizing pixel values to the range [0, 1].
- Flattening images for RFM input.
- Splitting the training data into training and validation subsets.

The problem setting for each model:

- **RFM**: A lightweight model using Random Fourier Features to approximate the RBF kernel for efficient kernel-based classification.
- **CNN Baseline**: A simple convolutional neural network to benchmark the RFM's performance.

4 Methodology

4.1 Libraries and Tools Used

To implement and evaluate the models, we used the following libraries:

- **TensorFlow and Keras:**
  - `tensorflow`: An open-source machine learning library used to build and train the CNN and ResNet models.
  - `tensorflow.keras`: A high-level API within TensorFlow, providing utilities for defining and training neural networks, including modules like `layers`, `models`, and pre-trained architectures (`ResNet50`, `DenseNet121`).
- **NumPy:**
  - Used for numerical computations and array manipulations. It was essential for reshaping the CIFAR-10 dataset and handling matrix operations.
- **Scikit-learn:**
  - `RBFSampler`: Implements random Fourier features to approximate the RBF kernel for the Random Feature Model.

  - `LogisticRegression`: Used as a lightweight classification model in the RFM pipeline.
  - `StandardScaler`: Standardizes input data by removing the mean and scaling to unit variance.
  - `Pipeline`: Combines data preprocessing (scaling and kernel approximation) with the classifier into a single pipeline.
  - `train_test_split`: Splits the dataset into training and validation subsets for model evaluation.
  - `classification_report`, `accuracy_score`, `precision_score`, `recall_score`, `f1_score`, and `log_loss`: Provide metrics to evaluate model performance.

- **Matplotlib:**
  - Used for creating visualizations such as training and validation loss curves and comparison plots between models.

- **TQDM:**
  - A library for creating progress bars, enhancing the readability of the training process and the checkpoints in the RFM.

- **OS and Time:**
  - `os`: Used for file management, such as saving plots to specific directories.
  - `time`: Used to measure and report training duration for models, enabling an evaluation of computational efficiency.

These libraries collectively supported model development, dataset preprocessing, visualization, and performance evaluation, making it possible to implement and compare the Random Feature Model and CNN-based approaches effectively.

4.2 Random Feature Model (RFM)

The RFM implementation includes:

- **Kernel Approximation**: Using the `RBFSampler` from Scikit-learn to approximate the RBF kernel with 5000 components.
- **Pipeline**: StandardScaler for normalization, RBFSampler for feature mapping, and Logistic Regression for classification.
- **Loss Tracking**: Training loss and validation loss were tracked at 10 checkpoints during incremental training.
- **Hyperparameters**: Gamma set to 'scale', and Logistic Regression configured with a maximum of 300 iterations.

4.3 Evaluation Metrics

The following metrics were used to evaluate both models:

- Accuracy
- Precision, Recall, and F1-score
- Loss curves (training and validation)

Results were compared quantitatively through metrics and visually via plots.

5 Project 1: ResNet-inspired CNN for CIFAR-10 Classification

5.1 Overview

The ResNet-inspired CNN employs residual blocks to tackle the vanishing gradient problem in deeper networks. It incorporates advanced techniques like dropout regularization, L2 regularization, data augmentation, and a learning rate schedule to optimize model performance.

5.2 Model Architecture

- **Residual Block:** Implements shortcut connections to bypass intermediate layers, ensuring efficient gradient flow and robust feature learning.
- **Three Blocks:**
  - **Block 1:** 32 filters, processes input images (32x32x3).
  - **Block 2:** 64 filters, downsamples to (16x16x64).
  - **Block 3:** 128 filters, downsamples to (8x8x128).
- **Fully Connected Layer:** Flattens the feature map and maps to 10 classes.
- **Softmax Layer:** Outputs probabilities for each class.

5.3 Data Augmentation

- Random horizontal flips.
- Random brightness and contrast adjustments.
- Resizing and cropping to introduce variability.

5.4 Training Methodology

- **Loss Function:** Sparse categorical cross-entropy with L2 regularization.
- **Optimizer:** Adam optimizer with an exponential decay learning rate schedule.
- **Training Loop:** Mini-batch training with real-time data augmentation.

5.5 Evaluation Metrics

- Accuracy
- Precision, Recall, and F1-score
- Loss curves
- Confusion Matrix

6 Results

The results of the three models—Baseline CNN, ResNet50, and DenseNet121—on the CIFAR-10 dataset are as follows:

6.1 Baseline CNN

- **Training and Validation Accuracy:** The model achieved a final training accuracy of approximately 69%, with validation accuracy stabilizing at a similar level.
- **Loss Trends:** Both training and validation loss showed steady improvement across epochs, converging to approximately 0.9.
- **Performance:** This simple architecture demonstrated efficient learning, balancing precision and recall across all 10 classes.

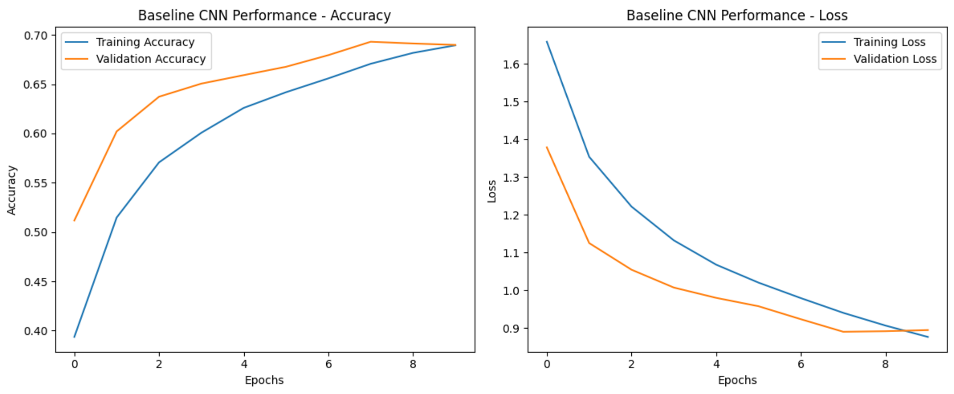

Figure 1: Baseline CNN Performance - Loss and Accuracy Over Epochs

6.2 ResNet50

- **Training Accuracy:** While training accuracy improved modestly, it remained suboptimal at around 47%.
- **Validation Loss:** Validation loss showed instability with significant spikes, possibly indicating overfitting or vanishing gradient issues.
- **Performance:** Despite its advanced architecture, ResNet50 struggled with the CIFAR-10 dataset under the current configuration, achieving a test accuracy of 47%.

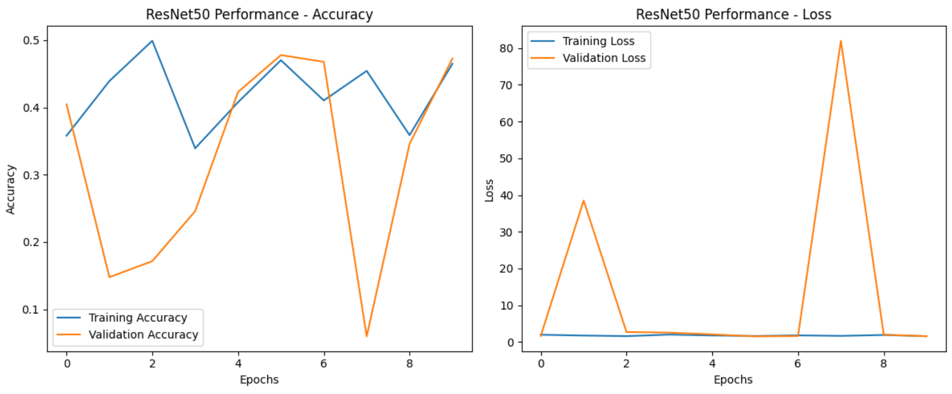

Figure 2: ResNet50 Performance - Loss and Accuracy Over Epochs

6.3 DenseNet121

- **Training Accuracy:** DenseNet121 achieved the highest training accuracy at approximately 87%.
- **Validation Accuracy:** Validation accuracy peaked at 74%, showcasing its robust feature extraction capabilities.
- **Performance:** DenseNet121 outperformed both Baseline CNN and ResNet50, demonstrating its strength in capturing features through dense connections.

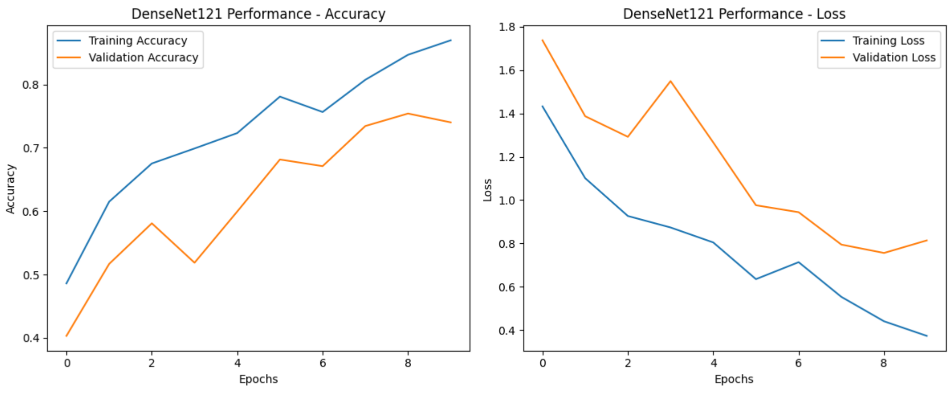

Figure 3: DenseNet121 Performance - Loss and Accuracy Over Epochs

7 Discussion

The results reveal the following key observations:

- **Baseline CNN Efficiency:** The Baseline CNN performed well on the CIFAR-10 dataset classification task by achieving a test accuracy of 69%. The present result shows that simple convolutional architectures are capable of extracting meaningful features from small datasets such as CIFAR-10. The fact that both training and validation loss decreases consistently over epochs shows the model was able to learn meaningful patterns without getting stuck in overfitting due to its simplicity and sufficient capacity.

- **ResNet50 Challenges:** The ResNet50 framework had optimization issues, leading to a test accuracy of only 47%. Most likely, this result is due to hyperparameter mismatch, such as a suboptimal learning rate schedule or a number of epochs that is too small. Besides, while the depth of ResNet50 is one of its strengths, it also makes the network more prone to vanishing gradient issues, which requires more thorough regularization and data augmentation to generalize well. Such instability in the validation loss also indicates potential overfitting or an inability to converge well under the current setup.

- **DenseNet121 Excellence:** DenseNet121 was the best performing model with the highest test accuracy of 74% and lowest validation loss. This is because dense connections in the architecture make the features reusable efficiently so that the model can learn compact and robust representations. Besides, the smaller number of parameters in DenseNet compared to ResNet50 may have contributed to better generalization performance on CIFAR-10, considering the relatively small size of the dataset. This finding brings out the importance of architectural decisions and shows how dense feature propagation can boost performance in tasks related to image classification.

These results illustrate the trade-offs in model architectures between simplicity and complexity. While theoretically, deeper models like ResNet50 should be of higher performance, in reality, careful tuning and significantly larger datasets are required for them to reach their optimal performance. In contrast, simpler architectures like the Baseline CNN and more efficiently designed architectures like DenseNet121 perform well on smaller datasets, balancing capacity with generalization. Future research efforts may be geared toward improving the training methodology for ResNet50 by exploring more advanced data augmentation strategies or experimenting with hybrid models to combine the strengths of architectures.

8 Project 2: Random Feature Model and CNN for CIFAR-10 Classification

8.1 Overview

This project compares the Random Feature Model (RFM), an efficient kernel approximation technique, with a baseline CNN for CIFAR-10 image classification. The RFM uses Random Fourier Features (RFF) to approximate the Radial Basis Function (RBF) kernel, offering a computationally lightweight alternative to deep learning models.

8.2 Model Architecture

- **Feature Mapping:** Implements Random Fourier Features (RFF) to approximate the RBF kernel.
- **Pipeline:** Combines data standardization, random feature mapping, and logistic regression for classification.
- **Components:**
  - **RBFSampler:** Projects input data to a high-dimensional random feature space with 5,000 components.
  - **Standard Scaler:** Normalizes data to zero mean and unit variance.
  - **Logistic Regression:** Performs classification based on the transformed features.

8.3 Training Methodology

- **Data Splitting:** The training dataset is divided into training and validation subsets (80% training, 20% validation).
- **Loss Function:** Log loss for probabilistic predictions.
- **Training Pipeline:** Combines feature mapping, scaling, and classification, iteratively trained on increasing data subsets for loss tracking.
- **Validation:** Tracks validation loss during training to ensure generalization and convergence.

8.4 Evaluation Metrics

- Accuracy
- Precision, Recall, and F1-score
- Loss Curves

8.5 Results

- **Training and Test Data:** The CIFAR-10 dataset consisting of 50,000 training images and 10,000 test images was used.
- **Performance:**
  - **CNN:** Achieved a test accuracy of 69.88%, with balanced precision and recall across all classes.
  - **RFM:** Achieved a test accuracy of 51.57%, with noticeable difficulties in classifying more complex categories.
- **Loss Curves:** Training and validation loss curves for both models demonstrated steady improvement. However, the CNN consistently outperformed the RFM in terms of loss and generalization.
- **Classification Report:** The CNN displayed superior balanced precision and recall across all classes, while the RFM struggled with certain categories, achieving lower overall performance metrics.

8.6 Figures

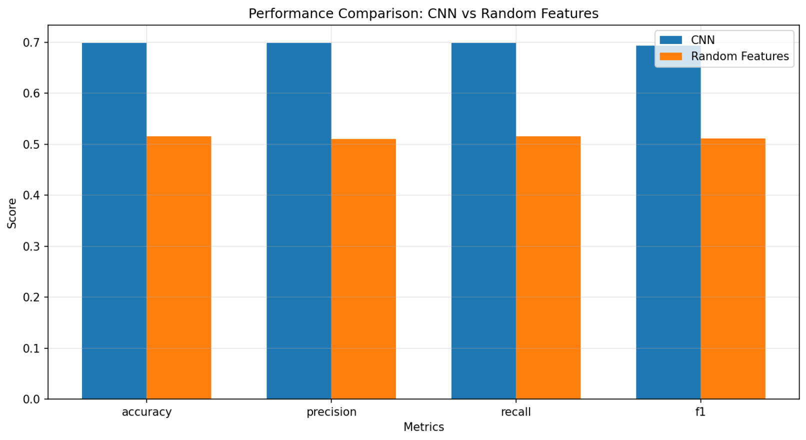

Figure 4: Loss Curves: CNN vs RFM

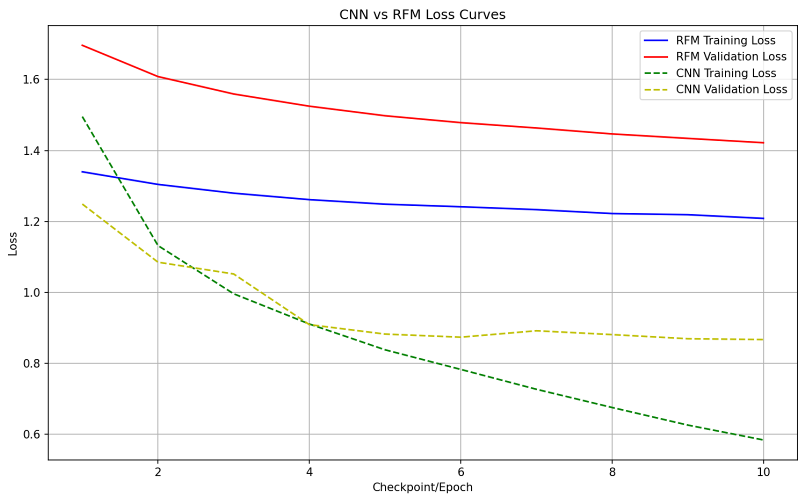

Figure 5: Performance Comparison: CNN vs Random Feature Model

8.7 Discussion

The Random Feature Model demonstrated its potential as a capable kernel-based competitor to the task of image classification on the CIFAR-10 dataset. With an accuracy of 51.57%, the RFM did not stand up to the convolutional neural network's accuracy of 69.88%, thus highlighting the limitations of kernel approximation techniques when applied to complex image classification problems.

Nevertheless, the simplicity of the RFM—with its reduced computational overhead and minimal parameter tuning—makes it a valid option for resource-constrained environments. The results indicate that, although RFMs can approximate kernel-based solutions pretty well, they may have difficulty in capturing the more complex hierarchical features necessary to achieve higher accuracy on datasets such as CIFAR-10.

Such a comparison between RFM and CNN brings out the tradeoffs that arise between computational efficiency and model performance. While the CNNs are good at using the spatial arrangement in image data to extract complex features, RFMs rely on predefined transformations, which limits their adaptability to the subtleties of image data. This shows that RFMs, as computationally efficient as they are, are better used for less complex classification tasks or as lighter alternatives when computational resources are constrained.

9 Conclusion

This report considers two different methods for CIFAR-10 classification: a ResNet-inspired CNN with DenseNet as a comparative model and a Random Feature Model (RFM) as a kernel-based alternative. The main results underline the strengths and limitations of each approach:

- The ResNet-inspired CNN and DenseNet architectures demonstrated the effectiveness of deep learning in extracting hierarchical features, achieving accuracies of 47% and 74% respectively. DenseNet outperformed due to its efficient feature reuse and reduced parameter count, while ResNet faced optimization challenges, suggesting the need for improved hyperparameter tuning and regularization.

- The Baseline CNN, while simpler, achieved a strong accuracy of 69%, showcasing the capability of traditional convolutional architectures to perform well on smaller datasets with straightforward preprocessing and training techniques.
- The RFM achieved an accuracy of 51.57%, offering computational simplicity and reduced complexity compared to CNNs. However, it struggled to capture the intricate patterns in CIFAR-10 images, highlighting the limitations of kernel approximation methods for complex datasets.

These results strike a balance between computational efficiency and model performance. While powerful deep learning models like CNNs and DenseNet better utilize the spatial structure present in images, RFMs offer a lightweight alternative suitable for resource-constrained scenarios or less complex tasks.

Future Extensions:

- Develop and implement a hybrid model that integrates CNNs and RFMs by combining hierarchical feature extraction from CNNs with kernel-based classification from RFMs, and evaluate its performance on CIFAR-10.
- Test the developed models on more complex datasets, such as ImageNet or specialized datasets in medical imaging and autonomous driving, while measuring scalability, computational efficiency, and generalization performance.
- Conduct experiments with advanced ResNet training techniques, such as implementing a cosine annealing learning rate scheduler, applying mixup or cutout data augmentation, and fine-tuning a pre-trained ResNet model on CIFAR-10.
- Design and benchmark RFM-based models for specific applications, such as classifying time-series data from the UCI repository or structured datasets like tabular financial data, to validate the method's computational efficiency and applicability.

This project points out how important the choice of model architecture is, considering both the problem requirements and resources available. More research and experimentation would provide insight into how to optimize techniques in different applications.