Convolutional Kernel Networks

1st Semester of 2024-2025

Podeanu Matei-Alexandru

Robert Eduard Schmidt

Abstract

Important to read: In this paper, we will present the architecture of convolutional kernel networks and illustrate how our implementation handles common classification tasks. We show how kernel-based methods can be structured in layers (much like classical convolutional neural networks) to learn feature representations. The goal is to investigate if such convolutional kernel networks can outperform or complement traditional CNNs on standard image classification benchmarks.

1 Introduction

In recent years, *convolutional neural networks* (CNNs) have become one of the most successful frameworks for a variety of classification tasks, particularly in computer vision. Despite their effectiveness, we became interested in *convolutional kernel networks* (CKNs)—a relatively new class of kernel-based models—to see whether they can provide advantages over or complementary insights to classical CNNs. Specifically, the primary problem we investigate in this paper is:

> *Can end-to-end kernel-based methods be effectively leveraged and trained to yield competitive or superior classification performance compared to traditional CNNs?*

In this section, we introduce our motivation and the key elements of our approach. We also summarize related works that have explored kernel methods, deep architectures, or both.

1.1 Contributions per Member

We outline below the specific contributions of each member of our team:

- **Matei**: Implemented the foundational CKN layers and optimized the kernel approximation techniques.
- **Robert**: Designed experiments to compare CKNs with classical CNNs on multiple benchmark datasets and analyzed their performance metrics.
- **Robert**: Conducted a thorough review of prior work on kernel-based methods, oversaw citation management.
- **Both**: Gathered the results of our tests and extracted the conclusions.

Summary of the Approach. In order to gain a comprehensive understanding of how convolutional kernel networks (CKNs) perform in practice, we identified five distinct classification tasks that we regarded as both challenging and representative of different data distributions. For each task, we implemented and fine-tuned our CKN-based model, working systematically to achieve the highest possible accuracy. Our aim was not only to evaluate the raw performance of CKN on a variety of benchmarks but also to explore how well the model generalizes, whether it can handle diverse input complexities, and how it compares to both classical convolutional neural networks and other state-of-the-art approaches.

1.2 Motivation and Significance

Although CNNs are widely successful, it is not fully established whether kernel-based layers could surpass or complement CNN performance on certain classification tasks.

2 Approach

The implementation for this project is available in the public repository at `https://github.com/M-Podi/Convolutional-Kernel-Networks`.

In our project, we employed a range of software tools including Python, PyTorch, Torchvision, and scikit-learn. Our approach centers around the use

of Convolutional Kernel Networks (CKNs) and standard convolutional neural networks (CNNs) combined with data preprocessing techniques such as normalization, data augmentation, and local contrast normalization (LCN). Below, we describe our methodology for each dataset.

2.1 MNIST

Preprocessing: The MNIST dataset is first loaded and transformed by converting the images to PyTorch tensors. A normalization transform is applied using the MNIST mean (0.1307) and standard deviation (0.3081) to standardize the data.

Model and Training: We employ a two-layer CKN model. In a layer-wise unsupervised pretraining phase, image patches are extracted and normalized, and K-means clustering is performed to initialize the filters for each layer. Once the filters are initialized, a classifier consisting of linear layers (with ReLU activations and dropout) is appended. The network is then trained using the cross-entropy loss and the AdamW optimizer. The evaluation on the test set yielded an accuracy of approximately 93.65

2.2 CIFAR-10

Two different approaches were implemented for CIFAR-10.

CKN-Based Approach:

- **Preprocessing:** The training and test images are converted to tensors and normalized using the CIFAR-10 channel means (0.4914, 0.4822, 0.4465) and standard deviations (0.2470, 0.2435, 0.2616). Data augmentation (e.g., random cropping or horizontal flipping) was also considered to improve robustness.

- **Model and Training:** Similar to the MNIST pipeline, patches are extracted from the images and normalized before applying K-means or spherical K-means clustering to initialize the filters in each CKN layer. A classifier is then trained using standard cross-entropy loss. One experimental run reported a test accuracy of around 30.97

CNN with Grad-CAM Visualization:

- **Preprocessing:** In this approach, the training transform includes random cropping, random horizontal flipping, and LCN.

- **Model and Training:** A simple CNN architecture is used with two convolutional layers (each followed by batch normalization, ReLU activation, and max pooling) and a fully connected classifier. The network is trained with the Adam optimizer and cross-entropy loss. The overall test accuracy achieved is approximately 74.47

2.3 SVHN (Street View House Numbers)

Preprocessing: For SVHN, the training data undergoes random cropping, random horizontal flipping, and LCN to enhance feature invariance and stabilize local pixel statistics.

Model and Training: A two-layer CKN model is employed, where the first layer uses raw images to extract patches and the second layer uses the output of the first layer as input. Patches are normalized and clustered using spherical K-means to initialize the filters. Following unsupervised filter initialization, the classifier is trained with an Adam optimizer and cross-entropy loss. The final test accuracy reported is approximately 82.80

2.4 Fruits 360

Two different pipelines were implemented for the Fruits 360 dataset:

CKN-Based Approach:

- **Preprocessing:** Images are resized to 32×32 pixels and preprocessed using LCN and random horizontal flipping.

- **Model and Training:** The CKN model is constructed in a similar manner as for the other datasets. Random patches are extracted from the training images, and spherical K-means clustering is applied sequentially for the two layers to initialize the filters. The classifier is trained using cross-entropy loss. This configuration achieved a test accuracy of about 88.97

Simple CNN Approach:

- **Preprocessing:** The images are processed using a similar pipeline (resizing, random horizontal flipping, ToTensor conversion, and LCN).

- **Model and Training:** A straightforward CNN architecture with two convolutional layers and a fully connected layer is trained in a supervised manner using the Adam optimizer

and cross-entropy loss. This approach attained a test accuracy of approximately 89.83

Evaluation: For each dataset, extensive evaluation is performed. In addition, for the CIFAR-10 approach, Grad-CAM is used to visualize and qualitatively assess the regions of the input image that most contribute to the classification decision.

In summary, our project leverages both unsupervised and supervised techniques with a focus on Convolutional Kernel Networks and CNNs. The integration of data augmentation, normalization (including LCN), and visualization via Grad-CAM enables a robust evaluation of the proposed methods across multiple datasets.

3 Limitations

Although our results indicate that convolutional kernel networks (CKNs) can perform competitively, several important limitations must be taken into account:

- **No publicly available implementation of the original paper:** The reference study that inspired our work did not provide open-source code or a detailed guide for its implementation. This lack of an official repository forced us to reconstruct many aspects of the algorithm from scratch, leaving possible gaps or discrepancies compared to the original version.
- **Difficulties reproducing the model's environment:** The article's methods relied on older Python versions and a platform that no longer hosts the relevant datasets. These dependencies complicated replication and testing, since we could not easily recreate the original environment or data conditions.
- **Limited computational resources:** Our experiments were primarily conducted on a modestly powered machine, restricting the size and depth of the networks we could train. Larger or more sophisticated architectures might yield better results but were impractical under our hardware constraints.
- **Incomplete mastery of mathematical concepts:** Certain theoretical points, such as advanced kernel approximations or deeper properties of reproducing kernel Hilbert spaces, remain partially understood by the team. This gap may have constrained both the interpretation of outcomes and the refinements of our approach.

4 Conclusions and Future Work

The CKN architecture achieved only a 2% performance improvement over a standard CNN, but with significantly longer training times due to the kernel layer training step. While Grad-CAM visualizations suggested that CKN might sometimes capture more meaningful features, these cases were too rare to draw strong conclusions.

A deeper analysis using a confusion matrix could have provided better insights into the model's strengths and weaknesses. Additionally, a more thorough exploration of Grad-CAMs or a reimplementation could highlight interesting patterns.

Overall, this was an engaging project that reinforced key concepts in data preprocessing, augmentation, and evaluation. Future projects could explore different architectures, model interpretability techniques, or real-world applications to enhance learning.

References

[1] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. *Learning Deep Features for Discriminative Localization*. arXiv preprint arXiv:1610.02391, 2016. Available: https://arxiv.org/abs/1610.02391

[2] LogB Research. *Understanding CKN: A Deep Dive into Convolutional Kernel Networks*. 2024. Available: https://logb-research.github.io/blog/2024/ckn/

[3] J. Mairal. *End-to-End Kernel Learning with Supervised Convolutional Kernel Networks*. Advances in Neural Information Processing Systems (NeurIPS), 2016. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/fc8001f834f6a5f0561080d134d53d29-Paper.pdf

[4] Claying. *CKN-Pytorch-image: Reference Implementation of Convolutional Kernel Networks*. 2024. Available: https://github.com/claying/CKN-Pytorch-image