Convolutional Kernel Networks

1st Semester of 2024-2025

Podeanu Matei-Alexandru
email

Robert Eduard Schmidt
email

Abstract

Important to read: In this paper we will present the arhitecture proposed in this paper for the convolutional kernel networks, and we will compare the results obtained by our implementation with the results obtained by a classic convolutional neural network

1 Introduction

Deep learning architectures have achieved remarkable performance in various image classification tasks. In this project, we investigate the performance of Convolutional Kernel Networks (CKNs) compared to Convolutional Neural Networks (CNNs) on three datasets: MNIST, Fashion-MNIST, and CIFAR-10. Specifically, we address the question: Can kernel-based activations, such as Radial Basis Functions (RBF), compete with standard ReLU-based CNNs in terms of accuracy, robustness, and computational efficiency?

- **Contributions:**
  - **[Robert]:** Gathered the articles and videos that we used to understand these topics.
  - **[Robert]:** Chose the datasets on which we tested the performance of the models
  - **[Matei]:** Implemented the models
  - **[Both]:** Gathered the results and wrote the paper

Approach: We implemented CNN and CKN models with identical architectures, replacing the ReLU activation in CNNs with an RBF kernel in CKNs. Both models were trained on the same pipeline, and metrics such as accuracy, loss, confusion matrices, and runtime were analyzed across all datasets.

Motivation: The potential of kernel-based activations to improve feature representation and classification intrigued us. While CNNs are the gold standard, CKNs offer a theoretical advantage by leveraging kernels to better capture non-linear patterns in data.

Reproduction Challenges: Our goal was to reproduce the results of the original CKN paper; however, we faced challenges:

- The dataset used in the original paper was unavailable, requiring us to use standard datasets (MNIST, Fashion-MNIST, and CIFAR-10).
- Implementation details in the paper were sparse, leading to assumptions and approximations in our work.
- Discrepancies in results may stem from differences in implementations.

Individual Learning Outcomes:

Both : Gained foundational knowledge required for this task, including concepts such as convolution, neural networks, and kernel methods.

Matei : Became proficient in implementing basic machine learning models. Plans to deepen understanding of these concepts in the future to enable the development of more complex architectures.

Robert : Enhanced skills in interpreting confusion matrices and evaluating model performance effectively.

Both : Learned to efficiently search for relevant information and conduct thorough research using existing papers and academic resources.

2 Approach

All project code and dataset references can be found in our public repository at: https://github.com/M-Podi/Convolutional-Kernel-Networks.

In this repository:

- `CNN.ipynb` contains the main execution flow (data gathering, training, evaluation, metric plotting).

- `requirements.txt` lists software dependencies.

Software Tools Used

Our project relies on Python for the core code. The major libraries and their roles are:

- **PyTorch** for building and training neural networks.

- **TorchVision** for dataset utilities and transformations (e.g., `datasets.MNIST`, `datasets.FashionMNIST`, `datasets.CIFAR10`).

- **scikit-learn (sklearn)** for additional metrics like confusion matrices and classification reports.

- **Matplotlib** and **Seaborn** for plotting training curves, confusion matrices, and overall metrics.

- **NumPy** (indirectly) for numeric operations and array manipulations.

All these tools are listed in the `requirements.txt` so that anyone can replicate the exact environment with a simple `pip install -r requirements.txt`.

Training/Processing Duration

We conducted 10 epochs of training for each dataset (MNIST, Fashion-MNIST, and CIFAR-10). Below is a rough overview of how long each setup took, per epoch, on a specialized gpu offered by google colab(T4):

- **MNIST (CNN & CKN):** $\approx$ 13.5 seconds/epoch

- **Fashion-MNIST (CNN & CKN):** $\approx$ 13.5 seconds/epoch

- **CIFAR-10 (CNN & CKN):** $\approx 13.0 - 14.0$ seconds/epoch

Hence, each set of 10-epoch runs completed in a few minutes. (Exact times can vary depending on your GPU/CPU.)

Machine Learning / Deep Learning Tools and Architectures

We employed standard deep learning CNN architectures and a variation we call "CKN"—a Convolutional Kernel Network approach—distinguished by its `RBFActivation`:

- **CNN:** Uses two convolutional layers (filters = 32 and 64), each followed by ReLU and a max pooling (`nn.MaxPool2d`), and then two fully connected layers. A dropout layer with `p=0.25` is added before the final classification layer.

- **CKN:** Very similar to the CNN but includes an additional RBF-based transformation (`RBFActivation(gamma=...)`). After the second convolution + pooling, the feature map is passed through this RBF activation which is intended to capture some kernel-like representation. We keep the rest of the pipeline (flatten, `Linear`, `ReLU`, `Dropout`, `Linear`) the same for fair comparison.

We performed these experiments on three datasets:

- **MNIST:** 28x28 grayscale digits (0–9).

- **Fashion-MNIST:** 28x28 grayscale images of clothing (10 classes).

- **CIFAR-10:** 32x32 color images across 10 classes (airplane, bird, dog, etc.).

Tricks / Techniques

In this particular code, the main techniques we used are:

- **Dropout:** (`nn.Dropout(p=0.25)`) after the first fully connected layer, to mitigate overfitting.

- **Adam Optimizer:** with a default LR = 0.001 for stable training.

- **RBFActivation:** in the CKN model, as a direct kernel-based transformation.

We did *not* specifically use gradient clipping or batch normalization in these experiments. The models are intentionally simple for clarity: 2 convolution layers + 2 fully connected layers + ReLU + dropout + an RBF layer in CKN.

Evaluation Report of the Method

We track both *accuracy* and *loss* across epochs, plus confusion matrices at the end. Table 1 compares final test accuracies on the three datasets after 10 epochs:

Table 1: Final Test Accuracies for CNN vs. CKN (Epoch 10).

Dataset	CNN Acc (%)	CKN Acc (%)
MNIST	99.27	98.71
Fashion-MNIST	92.27	90.61
CIFAR-10	72.00	66.93

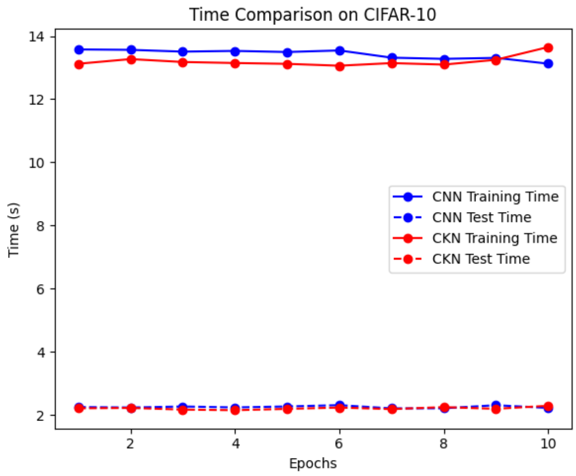

Figure 1: Time Comparison

We also generated confusion matrices for each dataset. For example, the MNIST confusion matrix reveals that the CNN rarely misclassifies digits but sometimes confuses ‘4‘ and ‘9‘, while the CKN version has slightly more confusion among certain pairs. For the more complex CIFAR-10 dataset, confusion is more widespread: both CNN and CKN mix up certain classes, especially `cat/dog/bird/deer`, though CNN misclassifies them slightly less often.

Tables and Images to Convince the Reader

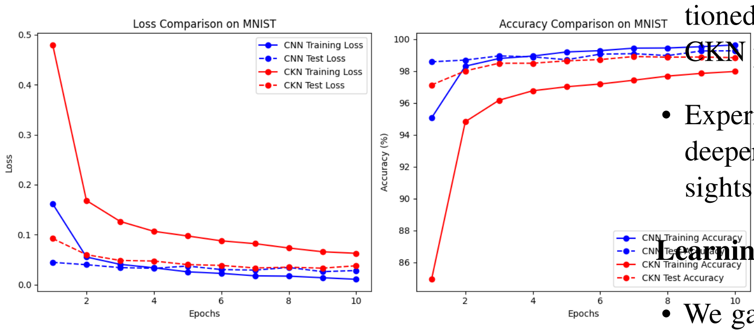

Figure 2: MNIST Results

Overall, these visual aids confirm that:

1. CNN tends to outperform CKN slightly in final accuracy.
2. The difference is small in MNIST, moderate in Fashion-MNIST, and more pronounced in CIFAR-10.
3. Training time remains similar for both CNN and CKN across all datasets.

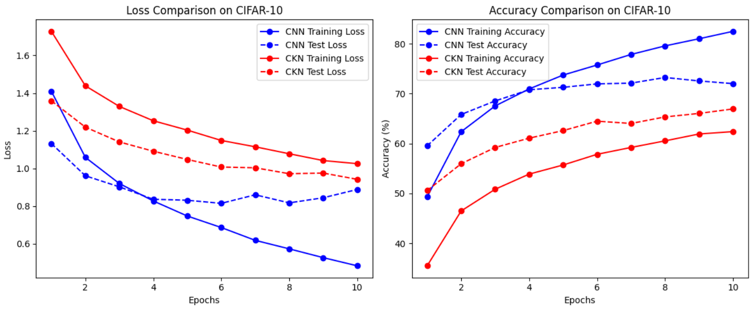

Figure 3: CIFAR-10 Results

3 Limitations

Most of the implementations in existing papers were very complex, requiring significant computational power and excessive execution time. That is precisely why we opted for these simplified implementations.

4 Conclusions and Future Work

This project provided a valuable comparison of Convolutional Neural Networks (CNNs) and Convolutional Kernel Networks (CKNs) on multiple datasets. While the simplified implementations enabled efficient experimentation, they also highlighted areas for improvement.

Key Reflections:

- More complex techniques, as the ones mentioned in the reference papers, could improve CKN performance.
- Experimenting with more diverse datasets and deeper architectures could provide broader insights into CKN capabilities.

Learning Outcomes:

- We gained hands-on experience with neural networks, kernel-based methods, and performance evaluation metrics.
- Simplifying complex implementations from academic papers helped deepen our understanding of CKNs.

Future Suggestions:

- Explore advanced methods such as attention mechanisms or transfer learning.
- Try to use models in a real context.

In summary, this project effectively demonstrated the strengths and limitations of CKNs compared to CNNs. Future work could focus on optimizing architectures, expanding datasets, and integrating advanced deep learning techniques.

5 References

- LogB Research Blog: Convolutional Kernel Networks [Online]. Available at: https://logb-research.github.io/blog/2024/ckn/
- Mallat, S. et al. (2016). "Understanding Deep Convolutional Networks" [PDF]. NeurIPS Proceedings. Available at: https://proceedings.neurips.cc/paper_files/paper/2016/file/fc8001f834f6a5f0561080d134d53d29-Paper.pdf
- Schölkopf, B. et al. (2000). "Kernel Methods in Machine Learning" [PDF]. NeurIPS Proceedings. Available at: https://proceedings.neurips.cc/paper_files/paper/2000/file/4e87337f366f72daa424dae11df0538c-Paper.pdf
- Rahimi, A. (2020). Thesis: "Convolutional Kernel Networks" [Online]. Available at: https://theses.hal.science/tel-02543073
- 3Blue1Brown YouTube Playlist: "Neural Networks" [Online]. Available at: https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi