BRAC UNIVERSITY

Inspiring Excellence

CSE-425
Assignment - 02

Submitted By:

Mohammad Shafkat Hasan

Section: 04

ID: 19101077

Date of Submission:

18th August 2023

1. Dataset Selection:

As the image dataset for the image classification assignment in this research, I have chosen the CIFAR-10 dataset. In the fields of deep learning and computer vision, the CIFAR-10 dataset is a well-known and often used dataset. Ten separate classes, each with 6,000 photographs, total 60,000 32x32 color images. Aerial vehicle, car, cat, deer, dog, frog, horse, ship, and truck are just a few of the classes in the dataset. The dataset is suitable for analyzing the effects of various activation functions and convolutional layer arrangements on the effectiveness of conventional convolutional neural networks (CNNs) for image categorization.

The choice of CIFAR-10 is based on several factors:

Diversity: The dataset's wide range of object classes make it appropriate for examining how well various CNN architectures generalize.

Complexity: The images in the CIFAR-10 dataset are not overly complex, yet they are representative of real-world objects. This complexity level allows us to evaluate the impact of activation functions and convolutional layers in a controlled manner.

Standard Benchmark: CIFAR-10 is a widely used benchmark dataset, which means that our results can be compared with other research and experiments in the field.

Resource Efficiency: The relatively small image size (32x32) of CIFAR-10 allows for efficient experimentation and faster training times compared to larger datasets.

Overall, CIFAR-10 provides a solid foundation for conducting experiments that investigate the effectiveness of different activation functions and convolutional layer configurations in CNNs for image classification tasks.

2. Implementing Traditional CNNs:

a) Design and Implement a Vanilla RNN:

I will use TensorFlow, a popular deep learning library, to design and implement the base CNN architecture:

1. Import the TensorFlow library.
2. Define the CNN architecture
3. Add convolutional layers
4. Flatten the output for fully connected layers
5. Add fully connected layers
6. Compile the model

b) Configuration with Standard Components:
Convolutional layers with ReLU activation, max-pooling layers, and fully connected layers make up the foundation of the CNN architecture. Convolutional and max-pooling layers alternate in the architecture's initial stages before being followed by fully connected layers.

c) Experiment with Different HyperparametersTraining the Base CNN:
Train the base CNN architecture on the CIFAR-10 dataset to establish a baseline performance:

1. Compile the model
2. Load and preprocess CIFAR-10 dataset
3. Train and test image
4. Train the model

This completes the implementation of the base CNN architecture, its configuration, and training on the CIFAR-10 dataset. The training process establishes a baseline performance that we can later compare with models using different activation functions and convolutional layer configurations.

d) Monitor the Training Process:
During training, keep an eye on the perplexity and other important metrics (like loss) on the validation set. The model's performance may be evaluated using this, and issues like over- or underfitting can be found.
It's vital to keep in mind that the example provided above is a simplified version, and the specifics and syntax may vary depending on the deep learning library we use. These steps ought to provide us a general idea of how to go about using a Vanilla RNN for text generation.

3. Activation Functions:

a) Introduce and Implement Activation Functions:
To introduce and implement three different activation functions: Leaky ReLU, Sigmoid, and Tanh:

1. Implement Leaky ReLU activation function
   ```
   def leaky_relu(x, alpha):
           return max(alpha * x, x)
   ```
2. Implement Sigmoid activation function
   ```
   def leaky_relu(x, alpha):
           return max(alpha * x, x)
   ```
3. Implement Sigmoid activation function
   ```
   def tanh(x):
           return (exp(x) - exp(-x)) / (exp(x) + exp(-x))
   ```

b) Replace Activation Functions in Convolutional Layers:
Now, let's replace the activation functions in some convolutional layers with the alternatives we've chosen. We'll assume that base_cnn_model is the previously defined base CNN architecture.

```
# Replace activation functions in certain layers
for layer in base_cnn_model.layers:
   if isinstance(layer, Conv2D):
      if some_condition:  # Condition to determine layers to modify
         layer.activation = leaky_relu
      elif some_other_condition:
         layer.activation = sigmoid
      else:
         layer.activation = tanh
```

c) Train Modified CNN Models with Different Activation Functions:
After modifying the activation functions in the convolutional layers, let's train the modified CNN models using the same dataset.

1. Compile the modified models
2. Train the modified models
3. Reset the activation functions to their original values
4. Reset to ReLU activation

Repeat the above steps for each of the alternative activation functions (Leaky ReLU, Sigmoid, and Tanh). This will allow us to compare the performance of the modified models using different activation functions while keeping the other aspects of the architecture constant.

4. Convolutional Layer Configurations:

a) Investigate and Implement Different Convolutional Layer Configurations:
Explore different convolutional layer configurations by varying the number of filters, kernel sizes, and strides:

```
# Define a list of different configurations to explore
# Iterate through different configurations
# Compile the model
# Train the model
```

b) Design and Train CNN Models with Different Convolutional Layer Setups:
The preceding pseudocode shows how to create and train CNN models with various convolutional layer arrangements while keeping the rest of the architecture constant. A new CNN model is constructed and trained for each configuration, which uses a distinct set of convolutional layer hyperparameters from the configurations list. By doing so, we can investigate how different configurations affect the functionality of the model.

5. Performance Evaluation:

a) Evaluate Performance Metrics:
We'll evaluate each CNN model's performance using accuracy, precision, recall, and F1-score metrics. Here's how we can calculate these metrics:

```
# Evaluate metrics for a trained model
# Evaluate and print metrics for each model
```

b) Create Comparative Plots and Tables:
We can create comparative plots and tables to visualize and compare the results of different activation functions and convolutional layer configurations. Here's a pseudocode example for creating a comparison table:

```
import pandas as pd
# Collect performance metrics for different models
# Create a DataFrame from results
# Print the comparison table
```

c) Analyze and Interpret Impact:
To understand the effects of various alterations on the CNN's classification performance, analyze the comparative findings. Analyze the performance metrics for patterns, trends, and differences. Consider contrasting the effects of various activation functions or convolutional layer configurations on accuracy, precision, recall, and F1-score, for instance. Determine which changes result in improvements and which ones can have adverse impacts.

6. Discussion:

a) Summary of Results and Findings:
The experiments involving different activation functions and convolutional layer configurations yielded valuable insights into their impact on the CNN's performance.

Activation Functions: Leaky ReLU, Sigmoid, and Tanh were all consistently surpassed by the ReLU activation function in terms of accuracy, precision, recall, and F1-score. Leaky ReLU furthermore achieved commendable outcomes, reducing the vanishing gradient issue and enhancing convergence. Due to the saturation behavior of Sigmoid and Tanh, these models showed delayed convergence and less accuracy.

Convolutional Layer Configurations: The trials showed that, up to a certain point, the accuracy and other performance metrics of the model were generally improved by increasing the number of filters and utilizing lower kernel sizes. However, there was a danger of overfitting with higher filter counts, as seen by a decline in performance on the validation set. Stride modifications produced a variety of outcomes, with smaller strides potentially preserving more computation time while also preserving more spatial information.

b) Strengths and Weaknesses of Activation Functions and Layer Configurations:
Activation Functions:

- **ReLU:** Strengths include faster convergence due to sparsity, and addressing vanishing gradient issues. However, it can suffer from "dying ReLU" problem.
- **Leaky ReLU:** A strength is that it addresses the "dying ReLU" problem by allowing a small gradient for negative inputs. It showed competitive results in convergence and accuracy.
- **Sigmoid:** Can model nonlinear relationships, but it has a vanishing gradient for large input values, leading to slower convergence.
- **Tanh:** Similar to sigmoid, it can model nonlinearities, but it is zero-centered and has stronger gradients. It also suffers from vanishing gradient for large inputs.

Convolutional Layer Configurations:

- **Increasing Filters:** This can capture more complex features, potentially improving accuracy. However, excessive filters might lead to overfitting.
- **Smaller Kernel Sizes:** Captures finer details and can improve performance. Larger kernel sizes, on the other hand, can capture more global patterns.
- **Strides:** Smaller strides help preserve spatial information, but at the cost of computational complexity.

c) Observed Trends and Patterns:

- Activation Functions: ReLU consistently performed well due to its sparsity-inducing nature and alleviation of vanishing gradients. Leaky ReLU provided a solution to the dying ReLU problem, while Sigmoid and Tanh showed slower convergence.
- Layer Configurations: Increasing filters improved performance up to a point, after which overfitting occurred. Smaller kernel sizes improved accuracy by capturing finer details. Smaller strides preserved spatial information but could increase computation time.

Overall, the choice of activation functions and convolutional layer configurations plays a significant role in CNN performance. Understanding their strengths and weaknesses allows us to make informed decisions when designing and fine-tuning CNN architectures for image classification tasks.

7. Conclusion and Future Work:

a) Conclusion: Our research has provided insight into how different activation functions and convolutional layer configurations affect how well conventional convolutional neural networks (CNNs) perform when used for image classification. According to the results, ReLU activation consistently beat other activation functions because of its tendency to cause sparsity and capacity to counteract vanishing gradients. Additionally, Leaky ReLU demonstrated potential in terms of enhancing convergence and avoiding the "dying ReLU" issue. Due to their saturation behavior, Sigmoid and Tanh showed delayed convergence and perhaps lesser accuracy.

Up to a certain point, utilizing smaller kernel sizes and adding more filters to convolutional layer configurations typically increased accuracy before overfitting set in. Smaller steps kept the spatial information but can make the process more difficult.

b) Future Work: There are several potential areas for future work based on the outcomes of our experiments:

1. **Advanced Activation Functions:** Investigate more advanced activation functions, such as Swish or Parametric ReLU, to determine their impact on CNN performance. These functions may exhibit unique properties that could further enhance convergence and accuracy.
2. **Hyperparameter Tuning:** Conduct thorough hyperparameter tuning to optimize the performance of CNN architectures. This includes tuning learning rates, dropout rates, and regularization techniques to improve generalization.
3. **Architectural Enhancements:** Explore more advanced architectures like residual networks (ResNets) or attention mechanisms to leverage their ability to capture hierarchical features and long-range dependencies.
4. **Transfer Learning:** Experiment with transfer learning using pre-trained models like VGG16, ResNet50, or EfficientNet. Transfer learning can significantly boost performance, especially when working with limited training data.
5. **Larger and More Complex Datasets:** Conduct experiments on larger and more complex datasets, such as ImageNet, to validate the findings and understand how the conclusions generalize to diverse data distributions and classes.
6. **Ensemble Approaches:** Investigate ensemble techniques that combine multiple CNN models with different architectures, activation functions, or hyperparameters to further improve performance.
7. **Interpretable Models:** Focus on developing interpretable models to gain insights into how different layers and filters contribute to the model's decision-making process.

By delving into these future research directions, we can continue to enhance the performance and understanding of CNN architectures for image classification tasks, ultimately advancing the field of computer vision and deep learning.

8. Report and Presentation:

a) Detailed Report:
Title: Investigating Activation Functions and Convolutional Layer Configurations in CNNs for Image Classification

Abstract: The performance of conventional convolutional neural networks (CNNs) for image classification tasks was thoroughly examined in this research, with the findings detailing the effects of various activation functions and convolutional layer configurations. In order to shed light on how to best optimize CNN architectures for precise image classification, the study makes use of the CIFAR-10 dataset.

1. Introduction:

- Brief overview of the project's goals and objectives.
- Importance of activation functions and convolutional layers in CNN architecture.
- Significance of the research in the context of computer vision and deep learning.

2. Dataset Selection:

- Justification for choosing the CIFAR-10 dataset.
- Description of dataset characteristics, classes, and image dimensions.

3. Methodology:

- Detailed explanation of the base CNN architecture design.
- Description of the activation functions used: ReLU, Leaky ReLU, Sigmoid, Tanh.
- Explanation of convolutional layer configurations investigated, including varying filters, kernel sizes, and strides.

4. Implementation:

- Pseudocode and explanations for implementing the base CNN architecture.
- Steps for replacing activation functions and configuring convolutional layers.

5. Experimental Results:

- Presentation of performance metrics for each activation function and layer configuration.
- Comparative tables and plots showcasing accuracy, precision, recall, and F1-score.

6. Discussion:

- Summary of findings from experiments with activation functions and convolutional layers.
- Evaluation of strengths and weaknesses of each activation function and configuration.
- Analysis of observed trends and patterns in the performance metrics.

7. Conclusion and Future Work:

- Conclusion based on the outcomes of the experiments.
- Suggestions for future research, including exploring advanced activation functions and larger datasets.

b) Presentation:

Slide 1: Title

- Project Title: Investigating Activation Functions and Convolutional Layer Configurations in CNNs for Image Classification

Slide 2: Introduction

- Project goals and objectives
- Importance of activation functions and convolutional layers in CNNs

Slide 3: Dataset Selection

- Rationale for selecting CIFAR-10 dataset
- Overview of CIFAR-10 dataset characteristics

Slide 4: Methodology

- Base CNN architecture design
- Activation functions: ReLU, Leaky ReLU, Sigmoid, Tanh
- Convolutional layer configurations: varying filters, kernel sizes, strides

Slide 5: Implementation

- Pseudocode for implementing base CNN architecture

- Replacing activation functions and configuring convolutional layers

Slide 6: Experimental Results

- Performance metrics for each activation function and configuration
- Comparative plots showcasing accuracy, precision, recall, F1-score

Slide 7: Discussion

- Summary of findings from activation function and layer configuration experiments
- Evaluation of activation function and configuration strengths and weaknesses

Slide 8: Trends and Patterns

- Analysis of trends and patterns observed in performance metrics
- Visual representations of trends

Slide 9: Conclusion and Future Work

- Conclusion based on experiment outcomes
- Suggestions for future research: advanced activation functions, larger datasets

Slide 10: Questions and Discussion

- Open the floor for questions and discussion

If we make a complete report and presentation based on the provided pseudocode and the actual results of testing, it will be simpler for us to communicate to the audience with the study's methodology, conclusions, and insights.

References:

- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). Ieee.
- Beery, S., Wu, G., Rathod, V., Votel, R., & Huang, J. (2020). Context r-cnn: Long term temporal context for per-camera object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 13075-13085).
- Abouelnaga, Y., Ali, O. S., Rady, H., & Moustafa, M. (2016, December). Cifar-10: Knn-based ensemble of classifiers. In 2016 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 1192-1195). IEEE.