Joseph Tschopp,

Final Project DIY CNN

05/02/2024

I. Introduction

Convolutional Neural Networks (CNNs) have become a fundamental component in the arsenal of machine learning technologies, with a pronounced efficacy in image recognition tasks. This project endeavors to evaluate the effectiveness of a DIY CNN on two benchmark datasets extensively recognized within the machine learning community: the MNIST dataset of handwritten digits and the Fashion-MNIST dataset of fashion items. These datasets are commonly employed to benchmark the performance of algorithms, especially in image classification.

The MNIST dataset is a collection of 60,000 training and 10,000 testing images, each depicting a single handwritten digit from 0 to 9. Each image is formatted as a 28x28 pixel grayscale representation, making it a staple dataset for entry-level applications in machine learning within computer vision. On the other hand, the Fashion-MNIST dataset was crafted as a more challenging counterpart to MNIST. It comprises 60,000 training and 10,000 testing images across ten classes of fashion articles, ranging from T-shirts and trousers to coats and shoes. Each item is similarly presented as a 28x28 pixel grayscale image, offering diverse images that introduce complexity in shape and form for classification tasks.

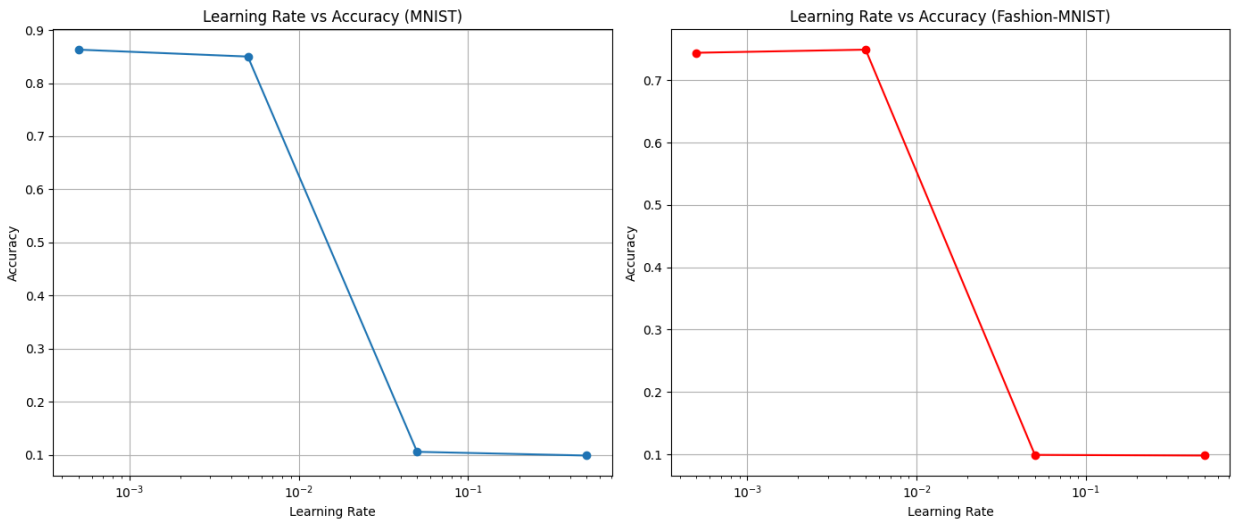
This report delves into the architecture of our custom-built CNN, outlines the training methodology employed, describes the experimental setup, and provides a comprehensive analysis of the model's performance across both datasets based on detailed accuracy and loss metrics observed over successive training epochs.

II. Methods

The CNN architecture developed for this study includes several layers, each designed to perform distinct functions necessary for effective image processing and classification. The initial layer of the CNN is a convolutional layer equipped with eight 3x3 filters. This layer is pivotal for detecting spatial patterns within the input images, such as edges and textures. Activation functions play a crucial role in neural networks by introducing non-linear properties to the model. Here, a Rectified Linear Unit (ReLU) activation function follows the convolutional layer to help the network learn complex patterns effectively.

Following the ReLU activation layer, a 2x2 max pooling layer is applied. The purpose of max pooling is to reduce the spatial dimensions of the feature maps, thus decreasing the computational load and the risk of overfitting by abstracting the input features. Subsequently, the pooled feature maps are flattened into a one-dimensional vector, serving as input for the densely connected layer at the network's end. This layer comprises ten neurons corresponding to the ten classes of the datasets and employs a softmax activation function to generate a probability distribution across the class labels.

Training the CNN involved using a cross-entropy loss function. This choice of loss function is apt for classification problems where the output can be interpreted as a probability value. Optimization was done using Stochastic Gradient Descent (SGD) with a learning rate of 0.005, which was selected after running experiments with different learning rates and seeing their corresponding accuracy.



Given the constraints in computational resources and model complexity, training was meticulously planned with a batch size of one. Although a small batch size can increase training time, it allows for very granular updates to the model weights, potentially leading to a more finely tuned model that can robustly generalize across new data.

III. Experimental Setup

Our experimental framework was built using Python, leveraging its robust libraries, such as NumPy, for numerical operations fundamental in executing neural network computations. The datasets were accessed and preprocessed through TensorFlow's Keras API, which simplifies the handling of these datasets by automating tasks like resizing the images and normalizing pixel values from a range of 0-255 to 0-1, thus preparing the data for processing by the CNN.

The practical execution of the model training was conducted via a specifically designed Python script (python cnn_mist_nonbatch.py). This script was responsible for initializing the CNN model, orchestrating the training process, and capturing the output metrics. The training was carried out sequentially for each image across ten epochs to facilitate meticulous adjustment of model weights based on the learning from each image. This method ensures thorough

learning, albeit at the cost of increased computational time. This approach also reproduces the experimental results for anyone with a Python setup, facilitating further verification and experimentation by peers or future studies.

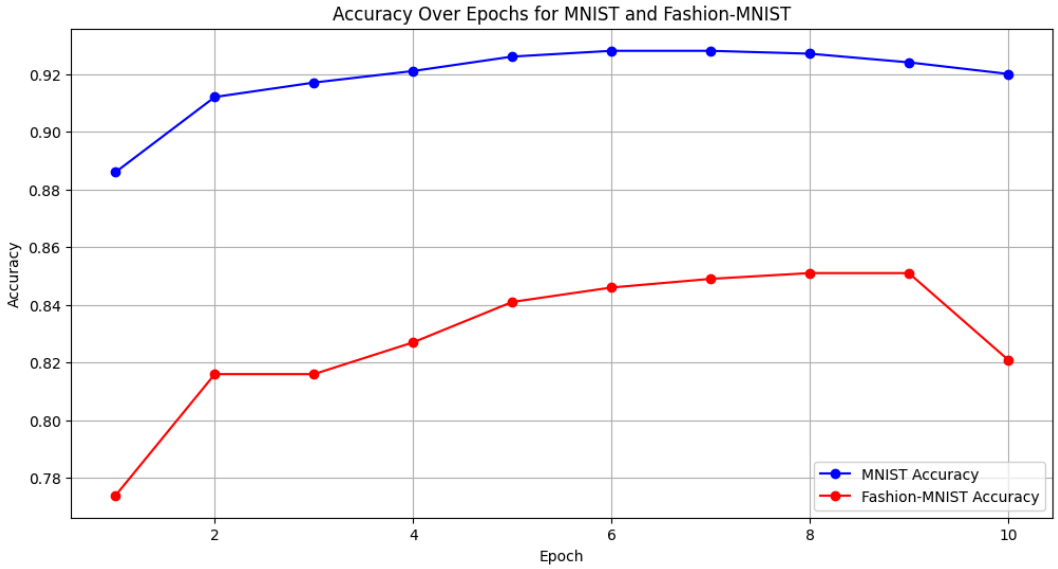
IV. Results

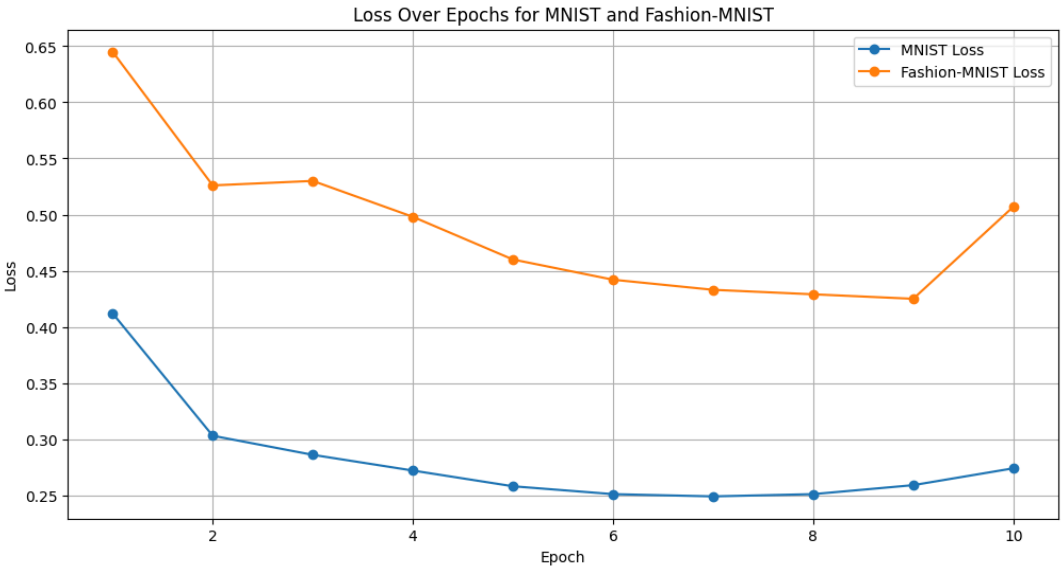
In this study, the performance of a DIY Convolutional Neural Network (CNN) was rigorously assessed through its training on the MNIST and Fashion-MNIST datasets over a series of 10 epochs. The results are graphically represented in the accompanying accuracy and loss plots, providing a clear visual insight into the model's performance dynamics throughout the training process.

The accuracy trajectory for the MNIST dataset exhibited a robust and steady improvement, achieving a peak accuracy of approximately 92% by the middle of the training period and maintaining a plateau after that. This indicates a successful learning outcome for the more straightforward MNIST dataset. In contrast, the accuracy for the Fashion-MNIST dataset showed a more gradual increase, achieving its highest at around 85% by the eighth epoch before experiencing a decline. This decline in later epochs suggests a potential overfitting issue, where the model might be too finely tuned to the training data, compromising its ability to generalize.

Correspondingly, the loss metrics from the training sessions reveal complementary trends. For MNIST, the loss decreased consistently, underscoring the accuracy improvements and signifying effective learning. Conversely, the loss values for Fashion-MNIST illustrated a more complex pattern; after initial decreases, the loss stabilized and ascended in the final epochs, aligning with the observed downturn in accuracy and supporting the inference of possible overfitting.

These visualizations underscore CNN's efficacy in handling the more straightforward MNIST data and highlight challenges faced with the more complex Fashion-MNIST data, such as enhanced generalization capabilities to prevent overfitting. The extended training time suggests that efficiency improvements could be garnered through optimization strategies, possibly involving adjustments to learning rates or the training duration. These findings are instrumental in guiding future modifications to the CNN architecture or its training protocol to optimize performance across varied datasets.





V. Discussion

The disparity in performance across the MNIST and Fashion-MNIST datasets can be attributed to several factors. The complexity and variability of Fashion-MNIST's images primarily make it a more demanding dataset, likely exacerbating the challenges in learning distinctive, generalizable features. This was reflected in the model's performance, which showed signs of overfitting, as evidenced by the drop in accuracy and increased loss in the later training epochs. No significant underfitting was observed, as the model initially demonstrated substantial learning capabilities in both datasets.

Several strategies could be adopted to enhance the model's performance, especially on complex datasets like Fashion-MNIST. Integrating more sophisticated architectural elements, such as additional convolutional layers or advanced activation functions, could help capture more complicated patterns. Regularization techniques like dropout or L2 regularization could mitigate the risk of overfitting by penalizing large weights. Data augmentation, such as image rotations, scaling, or color adjustments, could provide a richer set of training examples, promoting better generalization. Additionally, early stopping based on validation loss could prevent overtraining by halting the training process when no further improvement is observed, thereby conserving computational resources and preventing overfitting.

VI. Conclusion

In conclusion, while the DIY CNN demonstrated commendable performance on the more straightforward MNIST dataset, its efficacy on the more complex Fashion-MNIST dataset highlighted areas for potential enhancement. Future work could explore alternative CNN architectures or hybrid models incorporating elements of other successful deep-learning approaches to improve robustness and accuracy across varied datasets. This project contributes

to the ongoing exploration of CNN's capabilities. It provides a foundational analysis that can be built upon by future research to advance the field of image classification.