Giorgio Mendoza
CS549-E24-E01
Dr. Alexandros Lioulemes

Week 8 Lab

Part B_2:

numbersNN.py:

In this task, a basic neural network model was created using TensorFlow's Keras API to predict numerical values, demonstrating the principles of linear regression. The model consists of a single dense layer with one unit, designed to capture a linear relationship between a single input and output. It uses the stochastic gradient descent optimizer and mean squared error for loss calculation.

Training data comprised six defined pairs of inputs and outputs intended to model the linear equation `y = 2x - 1`. The model underwent 500 epochs of training, optimizing the weights to fit the provided data points accurately.

Post-training, the model was tested by predicting the output for an input value of `10.0`, aiming to evaluate its learning efficacy based on the theoretical output of `19.0` from the defined linear relationship. This exercise serves as a straightforward introduction to using neural networks for regression tasks, highlighting the process of model setup, training, and evaluation in TensorFlow and Keras. The result from the example is [[18.982384]]

fashionNN.py

In this section, a neural network model was constructed to classify images from the Fashion MNIST dataset, which includes various clothing items. The dataset was first loaded and then normalized to ensure pixel values were scaled between 0 and 1.

The designed model consisted of three layers: A Flatten layer to transform the 28x28 image matrices into a single vector of 784 elements. A dense layer with 128 nodes utilizing the ReLU activation function to introduce non-linearity into the model. A final dense layer with 10 nodes corresponding to the 10 different classes of clothing, using the softmax activation function to generate a probability distribution over the classes.

The model was compiled with the Adam optimizer, which adjusts learning rates throughout training, and used sparse categorical crossentropy as the loss function, suitable for multi-class classification of integer labels.

Training was conducted over five epochs, following which the model was evaluated against the test dataset to ascertain its accuracy. This provided a direct measure of how well the model generalizes to new, unseen data.

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.7827 - loss: 0.6233
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 948us/step - accuracy: 0.8623 - loss: 0.3809
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8785 - loss: 0.3379
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8880 - loss: 0.3070
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.8878 - loss: 0.3022
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 489us/step - accuracy: 0.8826 - loss: 0.3428
Test accuracy: 0.8795999884605408
```

The training process of the model showed significant improvements in accuracy over the course of five epochs, indicating effective learning. Initially, the model achieved an accuracy of 78.27%, which notably increased to 88.78% by the fifth epoch.

This progression reflects the model's increasing proficiency in classifying fashion items from the dataset. However, when evaluated against the test dataset, the accuracy slightly decreased to 88.26%, suggesting some overfitting to the training data but still indicating acceptable performance. This slight discrepancy highlights a common typical challenge in ML models, such as balancing between training accuracy and generalization to new data.

fashionCNN.py

The task involved utilizing the Fashion MNIST dataset to train a convolutional neural network aimed at classifying fashion products. The dataset was loaded and normalized to enhance the model's ability to learn effectively. Images were resized to include a color channel dimension, accommodating the CNN's input requirements.

The model architecture consisted of sequential layers designed to extract and learn from the spatial hierarchies in the image data. The network included three convolutional layers to identify features, paired with max pooling layers to reduce dimensionality and increase computational efficiency. Following the convolutional base, the model used a flattening step and dense layers to classify these features into one of ten fashion categories.

The model was compiled with the Adam optimizer and sparse categorical cross-entropy loss function. Training over five epochs showed rapid improvement in accuracy, demonstrating the model's effective learning from the training data. The final evaluation on the test dataset yielded an accuracy of approximately 90%, indicating positive performance and generalization from the model. This result showcases the CNN algorithm capability to handle and predict fashion item categories reliably from image data.

```
Epoch 1/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - accuracy: 0.7745 - loss: 0.6081

Epoch 2/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8858 - loss: 0.3083

Epoch 3/5
```

1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9092 - loss: 0.2506

Epoch 4/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9169 - loss: 0.2243

Epoch 5/5

1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9281 - loss: 0.1921

313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9013 - loss: 0.2859

Test accuracy: 0.9034000039100647

313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step

PaperRockSscissor.py

In this task, the goal was to develop a convolutional neural network to classify images into three categories (rock, paper, and scissors) using a labeled dataset. The process began by loading the dataset with TensorFlow Datasets, then the images were then normalized to scale the pixel values between 0 and 1, a step that helps improve the training dynamics.

A sequential CNN model was constructed, featuring multiple layers of convolutions and pooling followed by dropout and dense layers. This structure is designed to effectively extract features from the images and classify them accurately. The model was compiled using the Adam optimizer and categorical cross-entropy loss function, and it underwent training with the normalized images.

After training, the model's performance was evaluated on a test dataset. The training and validation accuracy metrics demonstrated close convergence, indicating that the model generalized well to new data without significant overfitting.

The final part of the task involved making predictions on new images to test the model's practical applicability. The high accuracy on both training and validation sets showcased the effectiveness of CNNs in handling visual recognition tasks.

1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 602ms/step
Predicted class: [1.000000e+00 5.381463e-08 8.670944e-21]

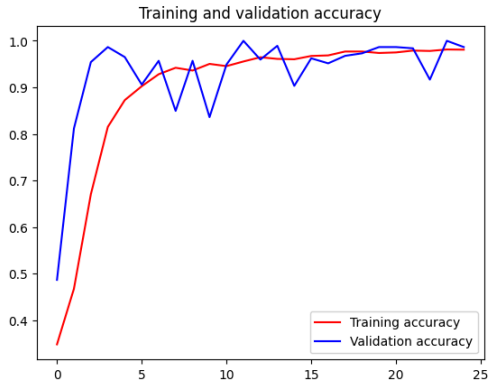

Figure 1. Training & Validation Accuracy

mnist.py

The objective of this exercise was to explore different optimizers for training a machine learning model using the MNIST dataset. The tutorial provided by TensorFlow guided the process of building, training, and evaluating a neural network model, testing the impacts of Adam, SGD, and RMSProp optimizers on model performance.

The script uses TensorFlow to load the MNIST dataset, normalize the image data, and then it constructs a Sequential model with dense layers for classification. The script defines functions for building the model, compiling and training with specified optimizers, and evaluating its performance on test data. It also includes functionality to plot training and validation accuracy for each optimizer to visually compare their performance.

The results demonstrated the effect of different optimizers on the training process. Each optimizer was evaluated based on its ability to minimize loss and maximize accuracy on both training and validation sets. The Adam optimizer showed rapid convergence and higher accuracy, while SGD displayed more gradual improvement over epochs. RMSProp, similar to Adam, facilitated quick learning but with slightly lower accuracy than Adam at the end of training.

The comparative plots clearly illustrate these differences, helping in understanding the trade-offs between convergence speed and model accuracy across different optimization algorithms.

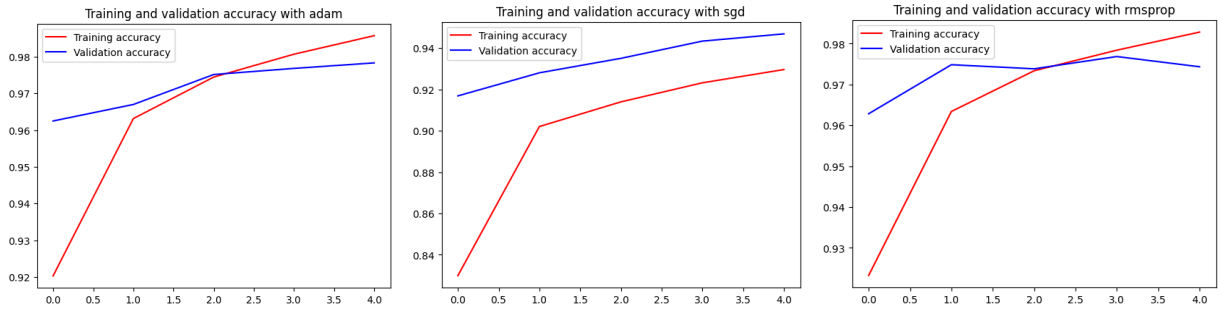

Figure 2. Training & validation using Adam, SGD and RMSPROP