Image Recognition of Handwritten digits with MNIST dataset

Summary:

Build and evaluate a convolutional neural network that classifies 28×28 MNIST digits (0–9) with > 97 % accuracy.

Sora Owada

Problem & Objectives

- **Why MNIST?**

    1. Standard benchmark for image recognition.
    2. Simple 10-class classification on 70k grayscale images.

- **Key Objectives:**

    1. Preprocess and normalize data to [0, 1] and reshape to (28, 28, 1).
    2. Define a CNN (Conv2D ↔ MaxPool → Dense).
    3. Train with Adam + sparse categorical cross-entropy.
    4. Track and visualize training/validation loss & accuracy.
    5. Evaluate on unseen test set and demonstrate sample predictions.

- **Sample Input:**

Data & Preprocessing

- **MNIST Dataset:**

  1. 60 000 train, 10 000 test, 28×28 px grayscale.
  2. Labels: 0–9 integer classes.

- **Preprocessing Steps:**

  1. **Normalize** pixel values to [0, 1].
  2. **Reshape** to (28, 28, 1) channel for CNN input.
     One-hot / sparse encoding for labels.

- **Code Snippet:**

```
# Load MNIST data
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

# normalize means all data would be in ragne between 0 to 1
x_train = tf.keras.utils.normalize(x_train, axis=1)
x_test = tf.keras.utils.normalize(x_test, axis=1)
```

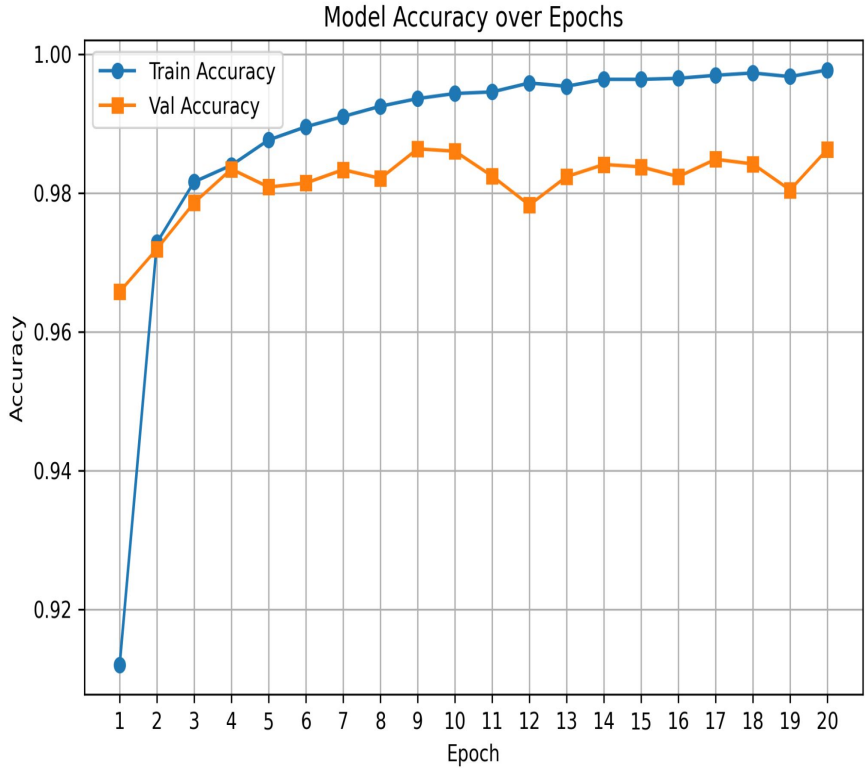
CNN Architecture & Training

```
# define model
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(32, (3, 3), strides = (1, 1), padding = 'same', activation = 'relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size = (2, 2), padding = 'same'))
model.add(tf.keras.layers.Conv2D(64, (3, 3), strides = (2, 2), padding = 'same', activation = 'relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size = (2, 2), padding = 'same'))
model.add(tf.keras.layers.Conv2D(128, (3, 3), strides = (2, 2), padding = 'same', activation = 'relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size = (2, 2), padding = 'valid'))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dense(64, activation='relu'))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
```

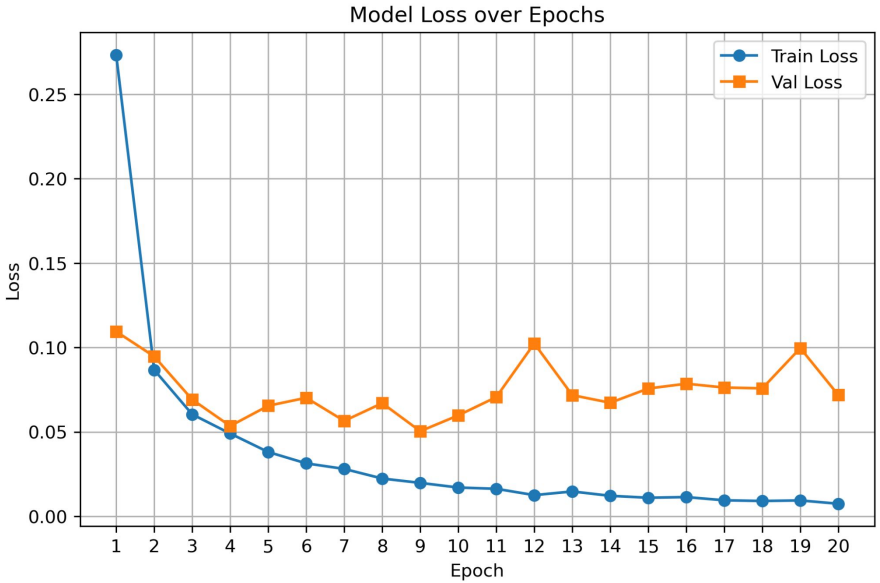
- **Conv2D:** Learns local patterns by convolving small trainable filters across the input.
- **MaxPooling2D:** Reduces each feature map's size by taking the maximum value over non-overlapping windows.
  Flatten: Converts the multi-dimensional feature maps into a single 1D vector.
- **Dense:** Fully connects inputs to outputs, combining all features to make final predictions.

Training Results

- **Accuracy over Epochs:**



- **Loss over Epochs:**



Reference

https://www.tensorflow.org/datasets/keras_example