MNIST Handwritten Digit Classifier Using JAX

🔍Project Title

MNIST Handwritten Digit Classifier Using JAX

📄Description

This is a simple neural network built from scratch using **JAX**, trained to recognize handwritten digits from the **MNIST** dataset (0–9).

- Framework: JAX
- Dataset: MNIST
- Model: 1 hidden layer neural network
- Purpose: Learn how neural networks work using forward pass, ReLU, softmax, loss, and gradient descent — all coded manually.

📖What is MNIST?

The **MNIST dataset** contains **70,000 grayscale images** of handwritten digits:

- 60,000 training images
- 10,000 testing images
- Each image: 28 × 28 pixels, values range from 0 to 255

How Neural Networks Work (with Forward Pass)

Step-by-step:

1. **Input Image**:
2. Each image is `28x28`, flattened into a **vector of 784 values** (pixels).
3. Normalized between 0 and 1 by dividing by 255.
4. **Weights Initialization**:
5. `W1`: (784 × 128) — connects input to hidden layer
6. `b1`: (128,) — bias for hidden layer
7. `W2`: (128 × 10) — connects hidden to output (10 digits)
8. `b2`: (10,) — bias for output

9. **Forward Pass:**

```
# Hidden Layer
z1 = X @ W1 + b1
a1 = relu(z1)

# Output Layer
z2 = a1 @ W2 + b2
y_hat = softmax(z2)
```

- `relu(x)`: ReLU (Rectified Linear Unit) turns negative values to 0.
- `softmax(x)`: Converts logits into probabilities across 10 classes.

- **Prediction**:

- The digit with the **highest probability** is the predicted class.

🧪 Loss Function

```
def cross_entropy(preds, labels):
    one_hot = jnp.eye(10)[labels]  # one-hot encode
    return -jnp.sum(one_hot * jnp.log(preds + 1e-7))
```

- Cross-entropy compares predicted probabilities to the actual label.
- Lower loss = better prediction.

🔦 Training Loop

1. **Loop for 1000 epochs**
2. Each step:
3. Pick a sample (`X_train[i]`)
4. Predict and calculate loss
5. Compute gradients using `jax.grad`
6. Update weights using simple gradient descent

```
grads = grad(loss_fn)(params, x, y)
params = {k: params[k] - lr * grads[k] for k in params}
```

1. Every 100 epochs: Print accuracy on test data

🔗Accuracy Output

Example:

```
Epoch 0, Accuracy: 0.1120
Epoch 100, Accuracy: 0.7945
Epoch 200, Accuracy: 0.8683
...
```

🛠️Requirements

```
pip install jax jaxlib tensorflow-datasets
```

📁File Overview

- `smalljaxproject.py`: Main script containing model, training, and evaluation
- `README.md`: Documentation (this file)

Author

Tejas S

🔮What to Try Next?

- Add batching instead of one sample per step
- Add more hidden layers
- Use `jax.jit` to speed up training
- Visualize weights and predictions
- Build a Gradio web interface