Google

MNIST:

An introductory example using NNX

Build and train a simple CNN with NNX

Download the MNIST dataset

```
#!wget -nc https://huggingface.co/datasets/ylecun/mnist/resolve/main/mnist/train-00000-of-00001.parquet
#!wget -nc https://huggingface.co/datasets/ylecun/mnist/resolve/main/mnist/test-00000-of-00001.parquet

import pandas as pd

train_file_path = '/content/train-00000-of-00001.parquet'
test_file_path = '/content/test-00000-of-00001.parquet'

mnist_train_df = pd.read_parquet(train_file_path)
mnist_test_df = pd.read_parquet(test_file_path)
```

Get the datasets ready

```
class Dataset:
    def __init__(self, df):
        self.df = df

    def __len__(self):
        return len(self.df)

    def __getitem__(self, index):
        return convert_to_numpy(self.df.iloc[index])

def convert_to_numpy(data_dict):
    png_bytes = data_dict['image']['bytes']
    image = Image.open(io.BytesIO(png_bytes))
    image_array = np.array(image, dtype=np.float32) / 255.0
    label_array = np.array(data_dict['label'])
    return {'image':image_array[:,:,np.newaxis], 'label':label_array}
```

Create the Grain dataloaders

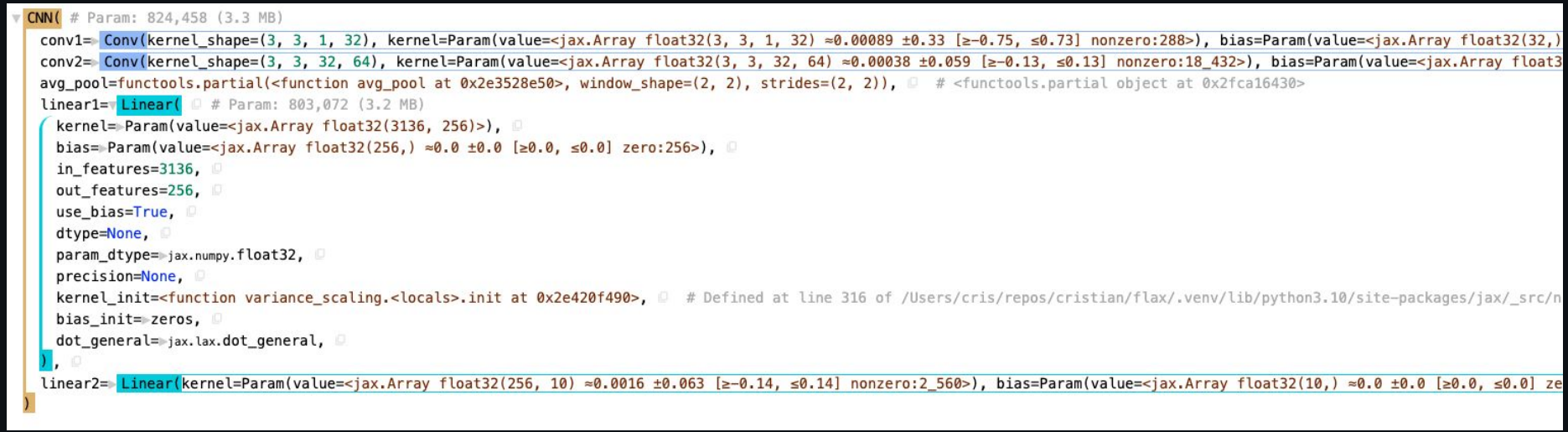
```
mnist_train = Dataset(mnist_train_df)
mnist_test = Dataset(mnist_test_df)

sampler = grain.SequentialSampler(
    num_records=len(mnist_train),
    shard_options=grain.NoSharding())

train_dl = grain.DataLoader(
    data_source=mnist_train,
    sampler=sampler,
    operations=[grain.Batch(batch_size=batch_size, drop_remainder=True)]
)

test_dl = grain.DataLoader(
    data_source=mnist_test,
    sampler=sampler,
    operations=[grain.Batch(batch_size=batch_size, drop_remainder=True)]
)
```

Define a CNN

```python
class CNN(nnx.Module):
  """A simple CNN model."""

  def __init__(self, *, rngs: nnx.Rngs):
    self.conv1 = nnx.Conv(1, 32, kernel_size=(3, 3), rngs=rngs)
    self.conv2 = nnx.Conv(32, 64, kernel_size=(3, 3), rngs=rngs)
    self.avg_pool = partial(nnx.avg_pool, window_shape=(2, 2), strides=(2, 2))
    self.linear1 = nnx.Linear(3136, 256, rngs=rngs)
    self.linear2 = nnx.Linear(256, 10, rngs=rngs)

  def __call__(self, x):
    x = self.avg_pool(nnx.relu(self.conv1(x)))
    x = self.avg_pool(nnx.relu(self.conv2(x)))
    x = x.reshape(x.shape[0], -1)  # flatten
    x = nnx.relu(self.linear1(x))
    x = self.linear2(x)
    return x
```

Display the CNN

```
# Instantiate the model.
model = CNN(rngs=nnx.Rngs(0))
# Visualize it.
nnx.display(model)
```



Test run the CNN

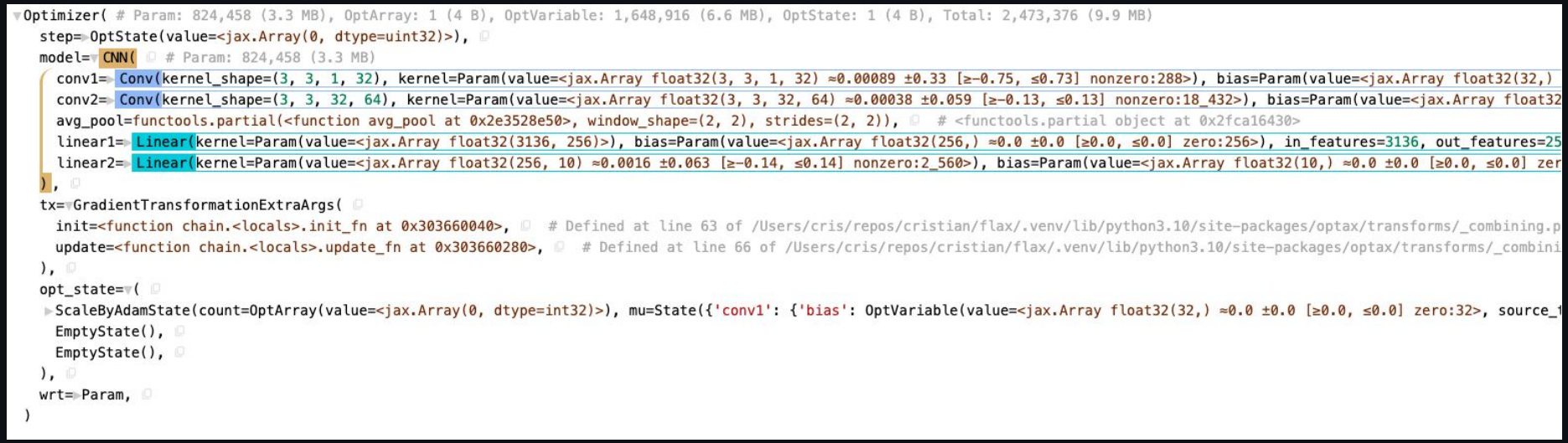
```
import jax.numpy as jnp  # JAX NumPy

y = model(jnp.ones((1, 28, 28, 1)))

# Array([[-0.06820839, -0.14743432,  0.00265857, -0.2173656 ,  0.16673787,
#         -0.00923921, -0.06636689,  0.28341877,  0.33754364, -0.20142877]],
#         dtype=float32)
```

Define optimizer and metrics

```
import optax

learning_rate = 0.005
momentum = 0.9

optimizer = nnx.Optimizer(model, optax.adamw(learning_rate, momentum), wrt=nnx.Param)
metrics = nnx.MultiMetric(
  accuracy=nnx.metrics.Accuracy(),
  loss=nnx.metrics.Average('loss'),
)
```

Define optimizer and metrics

```
nnx.display(optimizer)
```

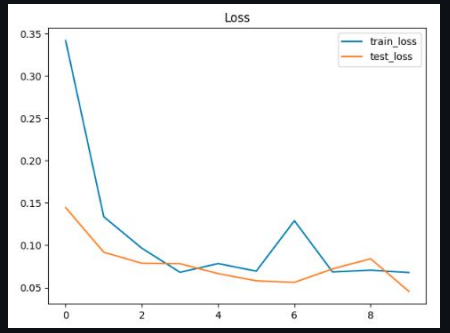


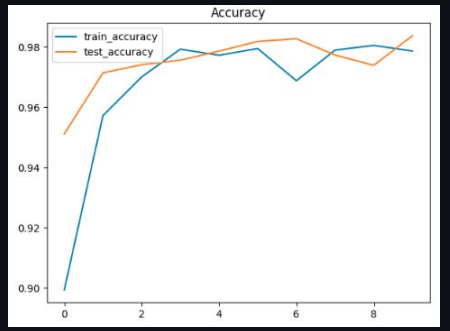
Define training step functions

```python
def loss_fn(model: CNN, batch):
  logits = model(batch['image'])
  loss = optax.softmax_cross_entropy_with_integer_labels(
    logits=logits, labels=batch['label']
  ).mean()
  return loss, logits

@nnx.jit
def train_step(model: CNN, optimizer: nnx.Optimizer, metrics: nnx.MultiMetric, batch):
  """Train for a single step."""
  grad_fn = nnx.value_and_grad(loss_fn, has_aux=True)
  (loss, logits), grads = grad_fn(model, batch)
  metrics.update(loss=loss, logits=logits, labels=batch['label'])  # In-place updates.
  optimizer.update(model, grads)  # In-place updates.

@nnx.jit
def eval_step(model: CNN, metrics: nnx.MultiMetric, batch):
  loss, logits = loss_fn(model, batch)
  metrics.update(loss=loss, logits=logits, labels=batch['label'])  # In-place updates.
```

Create the training loop

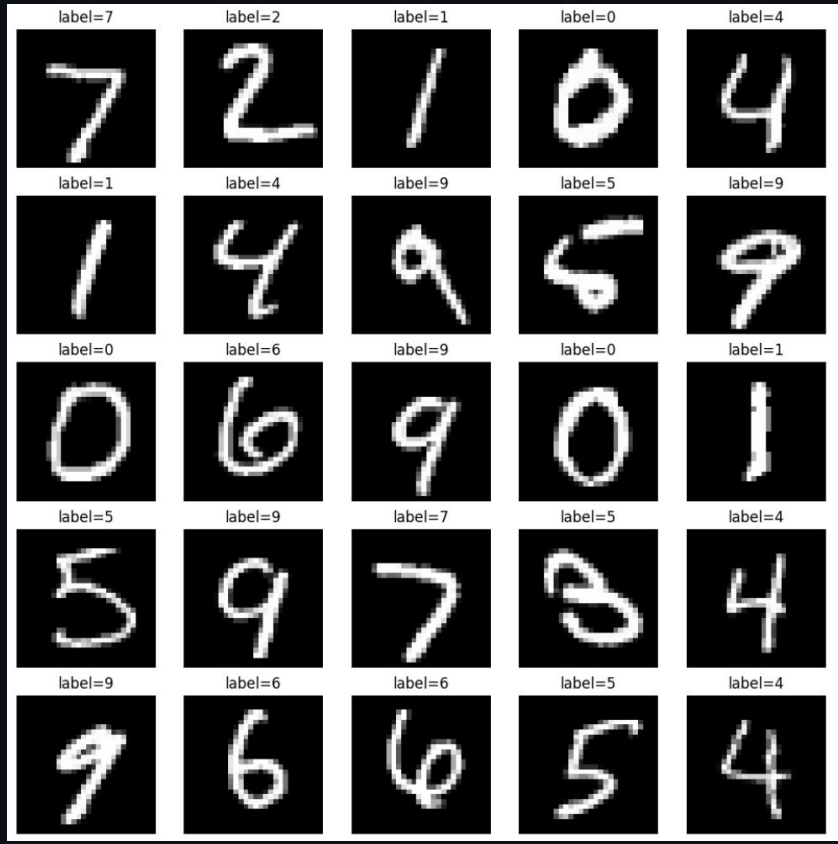
```
for epoch in range(num_epochs):
  for batch in train_dl:
    train_step(model, optimizer, metrics, batch)

    if step > 0 and (step % eval_every == 0 or step == train_steps - 1):
      for metric, value in metrics.compute().items():
        metrics_history[f'train_{metric}'].append(value)
      metrics.reset()  # Reset the metrics for the test set.

      for test_batch in test_dl:
        eval_step(model, metrics, test_batch)
```





Run a few test images

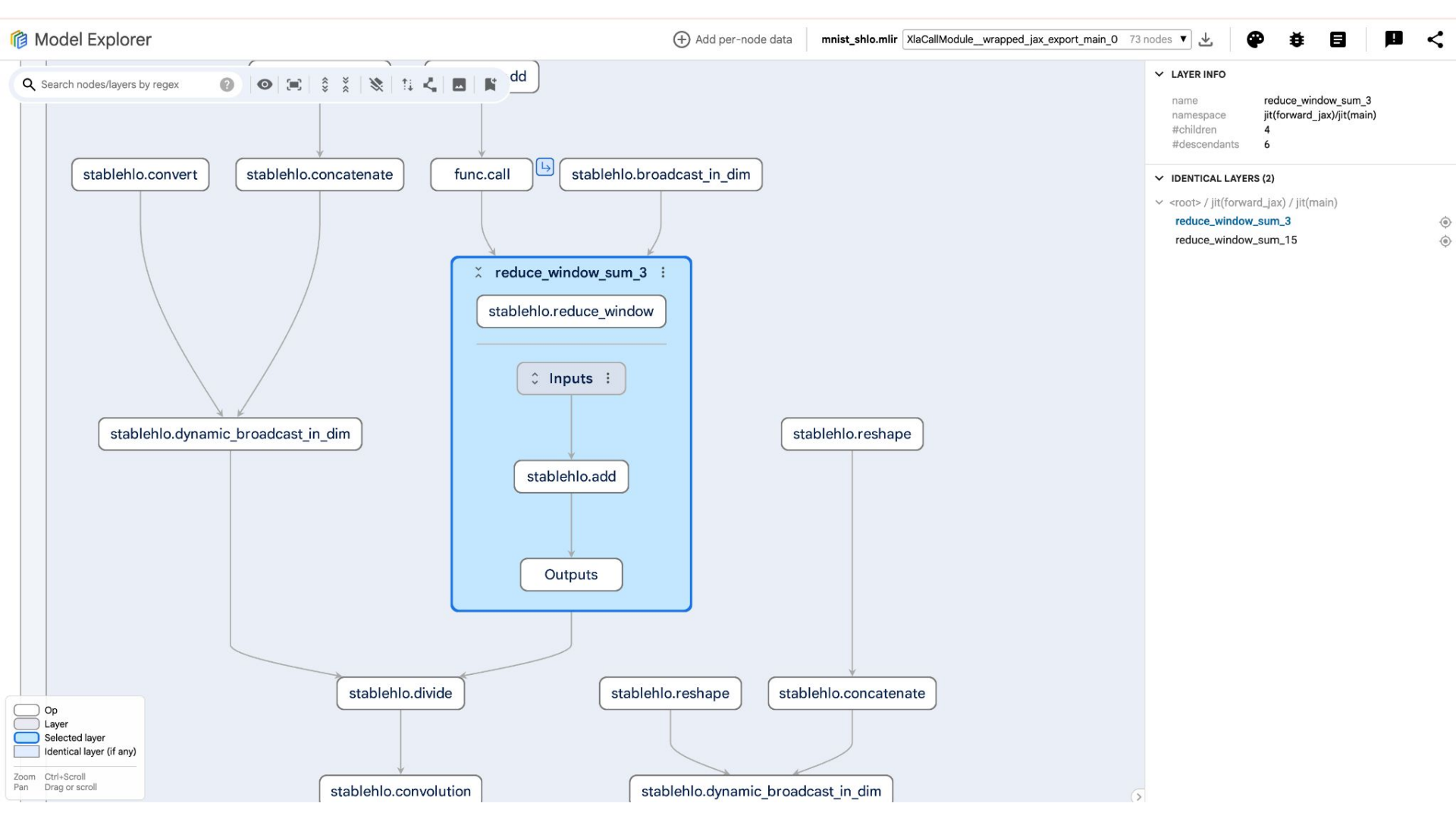
```
model.eval() # Switch to evaluation mode.

@nnx.jit
def pred_step(model: CNN, batch):
  logits = model(batch['image'])
  return logits.argmax(axis=1)

test_batch = next(iter(test_dl))
pred = pred_step(model, test_batch)

fig, axs = plt.subplots(5, 5, figsize=(12, 12))
for i, ax in enumerate(axs.flatten()):
  ax.imshow(test_batch['image'][i, ..., 0], cmap='gray')
  ax.set_title(f'label={pred[i]}')
  ax.axis('off')
```



Exploring the Model with Model Explorer

Model Explorer is a powerful graph visualization tool designed to help you understand and debug your ML models.

- Hierarchical Layout
  - View your model's structure clearly with nested layers that you can expand and collapse as needed.
- Metadata Overlays
  - Gain insights by overlaying metadata (e.g., attributes, inputs/outputs, etc) and custom data (e.g. performance) on nodes.
- Powerful Interactive Features
  - Focus on specific areas with search, navigate graphs smoothly with bookmarks and layer popups, customize node styles with queries, compare graphs side-by-side, and more.


Model Explorer
Add per-node data
mnist_shlo.mlir
XlaCallModule__wrapped_jax_export_main_0
73 nodes
Search nodes/layers by regex
stablehlo.convert
stablehlo.concatenate
func.call
stablehlo.broadcast_in_dim
reduce_window_sum_3
stablehlo.reduce_window
Inputs
stablehlo.add
Outputs
stablehlo.dynamic_broadcast_in_dim
stablehlo.reshape
stablehlo.divide
stablehlo.reshape
stablehlo.concatenate
stablehlo.convolution
stablehlo.dynamic_broadcast_in_dim
Op
Layer
Selected layer
Identical layer (if any)
Zoom Ctrl+Scroll
Pan Drag or scroll
LAYER INFO
name reduce_window_sum_3
namespace jit(forward_jax)/jit(main)
#children 4
#descendants 6
IDENTICAL LAYERS (2)
<root> / jit(forward_jax) / jit(main)
reduce_window_sum_3
reduce_window_sum_15

Learning Resources

Code Exercises, Quick References, and Slides

- https://goo.gle/learning-jax

Community and Docs

Community:

- https://goo.gle/jax-community

Docs

- JAX AI Stack: https://jaxstack.ai
- JAX: https://jax.dev
- Flax NNX: https://flax.readthedocs.io