simple_net_5_2

September 11, 2024

[2]:
```
pip install torch torchvision numpy pillow torchviz nbconvert
```

```
Collecting torch
  Downloading torch-2.4.1-cp39-cp39-manylinux2014_aarch64.whl (89.7 MB)
                             89.7/89.7 MB
4.4 MB/s eta 0:00:0000:0100:01
Collecting torchvision
  Downloading torchvision-0.19.1-cp39-cp39-manylinux2014_aarch64.whl (1.7 MB)
                             1.7/1.7 MB
4.4 MB/s eta 0:00:0000:0100:01
Collecting numpy
  Downloading
numpy-2.0.2-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl (13.9 MB)
                             13.9/13.9 MB
3.4 MB/s eta 0:00:0000:0100:01
Collecting pillow
  Downloading pillow-10.4.0-cp39-cp39-manylinux_2_28_aarch64.whl (4.4 MB)
                             4.4/4.4 MB
3.9 MB/s eta 0:00:0000:0100:01
Collecting torchviz
  Downloading torchviz-0.0.2.tar.gz (4.9 kB)
  Preparing metadata (setup.py) … done
Requirement already satisfied: nbconvert in /usr/local/lib/python3.9/site-
packages (7.16.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/site-packages
(from torch) (3.1.4)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.9/site-packages (from torch) (4.12.2)
Collecting filelock
  Downloading filelock-3.16.0-py3-none-any.whl (16 kB)
Collecting networkx
  Downloading networkx-3.2.1-py3-none-any.whl (1.6 MB)
                             1.6/1.6 MB
5.6 MB/s eta 0:00:00a 0:00:01
Collecting sympy
  Downloading sympy-1.13.2-py3-none-any.whl (6.2 MB)
                             6.2/6.2 MB
6.3 MB/s eta 0:00:0000:0100:01m
```

```
Collecting fsspec
  Downloading fsspec-2024.9.0-py3-none-any.whl (179 kB)
                             179.3/179.3
kB 3.6 MB/s eta 0:00:00a 0:00:01
Collecting graphviz
  Downloading graphviz-0.20.3-py3-none-any.whl (47 kB)
                             47.1/47.1 kB
3.7 MB/s eta 0:00:00
Requirement already satisfied: nbclient>=0.5.0 in
/usr/local/lib/python3.9/site-packages (from nbconvert) (0.10.0)
Requirement already satisfied: markupsafe>=2.0 in /usr/local/lib/python3.9/site-
packages (from nbconvert) (2.1.5)
Requirement already satisfied: mistune<4,>=2.0.3 in
/usr/local/lib/python3.9/site-packages (from nbconvert) (3.0.2)
Requirement already satisfied: importlib-metadata>=3.6 in
/usr/local/lib/python3.9/site-packages (from nbconvert) (8.4.0)
Requirement already satisfied: jupyter-core>=4.7 in
/usr/local/lib/python3.9/site-packages (from nbconvert) (5.7.2)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.9/site-
packages (from nbconvert) (1.3.0)
Requirement already satisfied: bleach!=5.0.0 in /usr/local/lib/python3.9/site-
packages (from nbconvert) (6.1.0)
Requirement already satisfied: nbformat>=5.7 in /usr/local/lib/python3.9/site-
packages (from nbconvert) (5.10.4)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.9/site-packages (from nbconvert) (1.5.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.9/site-
packages (from nbconvert) (24.1)
Requirement already satisfied: jupyterlab-pygments in
/usr/local/lib/python3.9/site-packages (from nbconvert) (0.3.0)
Requirement already satisfied: traitlets>=5.1 in /usr/local/lib/python3.9/site-
packages (from nbconvert) (5.14.3)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/site-
packages (from nbconvert) (0.7.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/site-
packages (from nbconvert) (4.12.3)
Requirement already satisfied: pygments>=2.4.1 in /usr/local/lib/python3.9/site-
packages (from nbconvert) (2.18.0)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.9/site-
packages (from bleach!=5.0.0->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.9/site-
packages (from bleach!=5.0.0->nbconvert) (0.5.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/site-
packages (from importlib-metadata>=3.6->nbconvert) (3.20.1)
Requirement already satisfied: platformdirs>=2.5 in
/usr/local/lib/python3.9/site-packages (from jupyter-core>=4.7->nbconvert)
(4.3.2)
```

```
Requirement already satisfied: jupyter-client>=6.1.12 in
/usr/local/lib/python3.9/site-packages (from nbclient>=0.5.0->nbconvert) (8.6.2)
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.9/site-
packages (from nbformat>=5.7->nbconvert) (4.23.0)
Requirement already satisfied: fastjsonschema>=2.15 in
/usr/local/lib/python3.9/site-packages (from nbformat>=5.7->nbconvert) (2.20.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.9/site-
packages (from beautifulsoup4->nbconvert) (2.6)
Collecting mpmath<1.4,>=1.1.0
  Downloading mpmath-1.3.0-py3-none-any.whl (536 kB)
                            536.2/536.2
kB 4.0 MB/s eta 0:00:00a 0:00:01
Requirement already satisfied: rpds-py>=0.7.1 in
/usr/local/lib/python3.9/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.20.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/usr/local/lib/python3.9/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (2023.12.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.9/site-
packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert) (24.2.0)
Requirement already satisfied: referencing>=0.28.4 in
/usr/local/lib/python3.9/site-packages (from
jsonschema>=2.6->nbformat>=5.7->nbconvert) (0.35.1)
Requirement already satisfied: pyzmq>=23.0 in /usr/local/lib/python3.9/site-
packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (26.2.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.9/site-packages (from jupyter-
client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.9.0.post0)
Requirement already satisfied: tornado>=6.2 in /usr/local/lib/python3.9/site-
packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (6.4.1)
Building wheels for collected packages: torchviz
  Building wheel for torchviz (setup.py) … done
  Created wheel for torchviz: filename=torchviz-0.0.2-py3-none-any.whl
size=4152
sha256=3ed91c6e8cfad40b0ad7523d8687ac1905ee55c6e9febe2d6138622a49e98636
  Stored in directory: /root/.cache/pip/wheels/29/65/6e/db2515eb1dc760fecd36b40d
54df65c1e18534013f1c037e2e
Successfully built torchviz
Installing collected packages: mpmath, sympy, pillow, numpy, networkx, graphviz,
fsspec, filelock, torch, torchviz, torchvision
Successfully installed filelock-3.16.0 fsspec-2024.9.0 graphviz-0.20.3
mpmath-1.3.0 networkx-3.2.1 numpy-2.0.2 pillow-10.4.0 sympy-1.13.2 torch-2.4.1
torchvision-0.19.1 torchviz-0.0.2
```

```
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager. It is recommended to
use a virtual environment instead: https://pip.pypa.io/warnings/venv

[notice] A new release of pip is
available: 23.0.1 -> 24.2
[notice] To update, run:
pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import time
import numpy


# Define transformations for the dataset
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

# Load the MNIST dataset
train_dataset = datasets.MNIST(root='./data', train=True, download=True,␣
 ↪transform=transform)
test_dataset = datasets.MNIST(root='./data', train=False, download=True,␣
 ↪transform=transform)

train_loader = DataLoader(dataset=train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(dataset=test_dataset, batch_size=64, shuffle=False)
```

[4]:
```
# SimpleNet modespecificl
class SimpleNet(nn.Module):
    def __init__(self):
        super(SimpleNet, self).__init__()
        self.conv1 = nn.Conv2d(1, 64, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(128, 128, kernel_size=3, padding=1)
        self.conv4 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
        self.conv5 = nn.Conv2d(256, 256, kernel_size=3, padding=1)
        self.conv6 = nn.Conv2d(256, 512, kernel_size=3, padding=1)
        self.conv7 = nn.Conv2d(512, 512, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
```

```
        self.fc1 = nn.Linear(512, 1024)
        self.fc2 = nn.Linear(1024, 10)

    def forward(self, x):
        x = self.pool(torch.relu(self.conv1(x)))
        x = self.pool(torch.relu(self.conv2(x)))
        x = torch.relu(self.conv3(x))
        x = self.pool(torch.relu(self.conv4(x)))
        x = torch.relu(self.conv5(x))
        x = self.pool(torch.relu(self.conv6(x)))
        x = torch.relu(self.conv7(x))

        x = x.view(x.size(0), -1)  # Flatten the feature map
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

[5]:
```
# Initialize the model, define the loss function and the optimizer
model = SimpleNet()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training loop with progress output and model saving
def train(model, train_loader, criterion, optimizer, epochs=5,␣
 ↪save_path='simple_net.pth'):
    model.train()  # Set model to training mode
    for epoch in range(epochs):
        start_time = time.time()
        running_loss = 0.0
        for batch_idx, (images, labels) in enumerate(train_loader):
            optimizer.zero_grad()  # Clear previous gradients
            outputs = model(images)  # Forward pass
            loss = criterion(outputs, labels)  # Compute loss
            loss.backward()  # Backward pass
            optimizer.step()  # Update weights

            running_loss += loss.item()

            if batch_idx % 100 == 0:  # Print progress every 100 batches
                print(f"Epoch [{epoch+1}/{epochs}], Batch [{batch_idx}/
 ↪{len(train_loader)}], Loss: {loss.item():.4f}")

        epoch_time = time.time() - start_time
        print(f"Epoch [{epoch+1}/{epochs}], Loss: {running_loss/
 ↪len(train_loader):.4f}, Time: {epoch_time:.2f} seconds")

    # Save the model
```

```
    torch.save(model.state_dict(), save_path)
    print(f"Model saved to {save_path}")
```

[6]:
```
# Test loop after loading the saved model
def test(model, test_loader, save_path='simple_net.pth'):
    # Load the saved model
    model.load_state_dict(torch.load(save_path))
    model.eval()  # Set model to evaluation mode
    correct = 0
    total = 0
    with torch.no_grad():
        for images, labels in test_loader:
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    print(f'Accuracy: {accuracy:.2f}%')
```

[5]:
```
# Train the model
train(model, train_loader, criterion, optimizer)
```

```
Epoch [1/5], Batch [0/938], Loss: 2.3011
Epoch [1/5], Batch [100/938], Loss: 0.6268
Epoch [1/5], Batch [200/938], Loss: 0.2710
Epoch [1/5], Batch [300/938], Loss: 0.0516
Epoch [1/5], Batch [400/938], Loss: 0.0427
Epoch [1/5], Batch [500/938], Loss: 0.0240
Epoch [1/5], Batch [600/938], Loss: 0.1251
Epoch [1/5], Batch [700/938], Loss: 0.0288
Epoch [1/5], Batch [800/938], Loss: 0.1199
Epoch [1/5], Batch [900/938], Loss: 0.0774
Epoch [1/5], Loss: 0.2907, Time: 582.59 seconds
Epoch [2/5], Batch [0/938], Loss: 0.0524
Epoch [2/5], Batch [100/938], Loss: 0.0464
Epoch [2/5], Batch [200/938], Loss: 0.0666
Epoch [2/5], Batch [300/938], Loss: 0.0110
Epoch [2/5], Batch [400/938], Loss: 0.0172
Epoch [2/5], Batch [500/938], Loss: 0.0463
Epoch [2/5], Batch [600/938], Loss: 0.1779
Epoch [2/5], Batch [700/938], Loss: 0.0129
Epoch [2/5], Batch [800/938], Loss: 0.0183
Epoch [2/5], Batch [900/938], Loss: 0.0649
Epoch [2/5], Loss: 0.0626, Time: 441.74 seconds
Epoch [3/5], Batch [0/938], Loss: 0.0609
Epoch [3/5], Batch [100/938], Loss: 0.1098
Epoch [3/5], Batch [200/938], Loss: 0.0019
```

```
Epoch [3/5], Batch [300/938], Loss: 0.1326
Epoch [3/5], Batch [400/938], Loss: 0.0262
Epoch [3/5], Batch [500/938], Loss: 0.1122
Epoch [3/5], Batch [600/938], Loss: 0.0045
Epoch [3/5], Batch [700/938], Loss: 0.0156
Epoch [3/5], Batch [800/938], Loss: 0.0111
Epoch [3/5], Batch [900/938], Loss: 0.0194
Epoch [3/5], Loss: 0.0492, Time: 535.25 seconds
Epoch [4/5], Batch [0/938], Loss: 0.1017
Epoch [4/5], Batch [100/938], Loss: 0.0010
Epoch [4/5], Batch [200/938], Loss: 0.0914
Epoch [4/5], Batch [300/938], Loss: 0.0231
Epoch [4/5], Batch [400/938], Loss: 0.0020
Epoch [4/5], Batch [500/938], Loss: 0.0861
Epoch [4/5], Batch [600/938], Loss: 0.0025
Epoch [4/5], Batch [700/938], Loss: 0.1879
Epoch [4/5], Batch [800/938], Loss: 0.0106
Epoch [4/5], Batch [900/938], Loss: 0.0023
Epoch [4/5], Loss: 0.0385, Time: 429.59 seconds
Epoch [5/5], Batch [0/938], Loss: 0.0314
Epoch [5/5], Batch [100/938], Loss: 0.0149
Epoch [5/5], Batch [200/938], Loss: 0.0000
Epoch [5/5], Batch [300/938], Loss: 0.0042
Epoch [5/5], Batch [400/938], Loss: 0.0103
Epoch [5/5], Batch [500/938], Loss: 0.0014
Epoch [5/5], Batch [600/938], Loss: 0.0020
Epoch [5/5], Batch [700/938], Loss: 0.0168
Epoch [5/5], Batch [800/938], Loss: 0.0139
Epoch [5/5], Batch [900/938], Loss: 0.0247
Epoch [5/5], Loss: 0.0341, Time: 434.97 seconds
Model saved to simple_net.pth
```

[7]:
```
# Test the model by reloading it
test(model, test_loader)
```

```
/tmp/ipykernel_22/3840639354.py:4: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
```

```
    model.load_state_dict(torch.load(save_path))
```

```
Accuracy: 98.99%
```

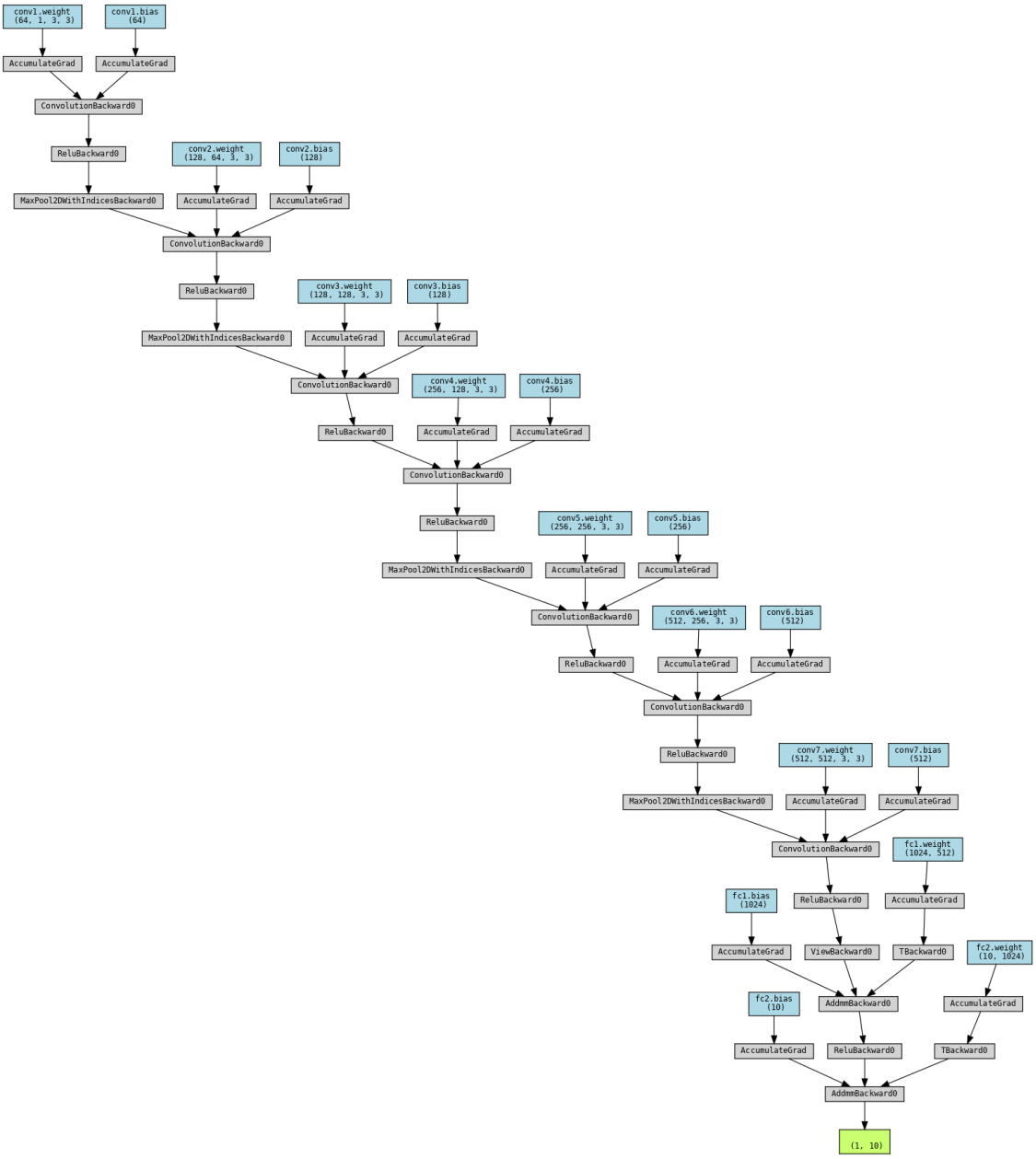
[8]:
```
from torchviz import make_dot
import torch
from IPython.display import Image, display

# Use a dummy input to visualize the model's architecture
x = torch.randn(1, 1, 28, 28)
y = model(x)

# Generate the architecture graph
dot = make_dot(y, params=dict(model.named_parameters()))

# Render the diagram as a PNG and display it inline in the notebook
dot.format = 'png'
dot.render('model_architecture')

# Display the image in the Jupyter notebook
display(Image('model_architecture.png'))
```



[13]:
```python
# Use only a subset of the layers for a simpler visualization
class SimpleNetPruned(nn.Module):
    def __init__(self):
        super(SimpleNetPruned, self).__init__()
        self.conv1 = nn.Conv2d(1, 64, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
```

```
        # We'll initialize fc1 after determining the output size of the␣
 ↪convolutional layers
        self.fc1 = None  # Placeholder, will be initialized later
        self.fc2 = nn.Linear(10, 10)  # Modify as needed for your final output

    def forward(self, x):
        x = torch.relu(self.conv1(x))
        x = self.pool(torch.relu(self.conv2(x)))

        if self.fc1 is None:
            # Dynamically calculate the input size for fc1
            num_features = x.view(x.size(0), -1).size(1)
            self.fc1 = nn.Linear(num_features, 10).to(x.device)

        x = x.view(x.size(0), -1)  # Flatten the feature map
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Create a simplified model
model_pruned = SimpleNetPruned()

# Use a dummy input to visualize the pruned model's architecture
x = torch.randn(1, 1, 28, 28)
y = model_pruned(x)

# Generate the architecture graph
dot = make_dot(y, params=dict(model_pruned.named_parameters()))

# Save the diagram as an SVG or PNG image
dot.format = 'png'
dot.render('model_architecture_pruned')
```

[13]:
```
'model_architecture_pruned.png'
```

[14]:
```
from torchviz import make_dot
import torch

# Assuming SimpleNet is already defined
model = SimpleNet()

# Use a dummy input to pass through the model
x = torch.randn(1, 1, 28, 28)
y = model(x)

# Generate the architecture graph
dot = make_dot(y, params=dict(model.named_parameters()))
```

```
# Save the diagram as an SVG or PNG image, SVG can be resized easily
dot.format = 'png'
dot.render('simplenet_architecture')
```

[14]: 'simplenet_architecture.png'

[ ]:

[ ]:

[ ]: