In [1]:
```
import sagemaker
from sagemaker.pytorch import PyTorch
import os
import pandas as pd
import random
from io import StringIO
import s3fs
import boto3

# Set up SageMaker session and role
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()
```

```
/opt/conda/lib/python3.11/site-packages/pydantic/_internal/_fields.py:198: Use
rWarning: Field name "json" in "MonitoringDatasetFormat" shadows an attribute
in parent "Base"
  warnings.warn(
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sage
maker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/sagemak
er-user/.config/sagemaker/config.yaml
```

In [2]:
```
s3_input_path = 's3://sudokudata/sudoku.csv'
```

Training Sample

In [3]:
```
sample_size = 1_000_000

# Open S3 file for streaming
fs = s3fs.S3FileSystem(anon=False)
with fs.open(s3_input_path, 'r') as f:
    header = next(f)
    # Initialize reservoir with the first k lines
    reservoir = [next(f) for _ in range(sample_size)]
    for i, line in enumerate(f, start=sample_size + 1):
        j = random.randint(1, i)
        if j <= sample_size:
            reservoir[j-1] = line

# Combine header and sampled lines
sampled_csv = header + ''.join(reservoir)

# Read into pandas DataFrame
df_sample = pd.read_csv(StringIO(sampled_csv))
print(df_sample.shape)
df_sample.head()
```

```
(1000000, 2)
```

Out[3]:

	puzzle	
0	2084900006000005499546000820670092003900100060...	27849516363178254995463178216785
1	7208650190050204700900008020486009200710025002...	72486531918532947639674185254865
2	0006050094090000005609738420000500018270905001...	78264513943981275656197384234671
3	8071003404095038211002000769703206002148005390...	82719634546957382115328497697832
4	0002079601296800540765000000010608970407590310...	45821796312968375437659421853146

In [5]:
```
df_sample.to_csv('sudoku_sampled_1M.csv', index=False)
s3 = boto3.client('s3')
s3.upload_file('sudoku_sampled_1M.csv', 'sudokudata', 'sudoku_sampled_1M.csv')
```

In [2]:
```
estimator = PyTorch(
    entry_point='train.py',
    source_dir='.',  # directory where train.py lives
    role=role,
    instance_type='ml.g4dn.xlarge',  # or 'ml.g4dn.xlarge'
    instance_count=1,
    framework_version='1.13',
    py_version='py39',
    hyperparameters={
        'epochs': 5,
        'batch-size': 64,
        'lr': 0.001
    },
    output_path=f's3://{sagemaker_session.default_bucket()}/sudoku-model-outpu
)
```

In [3]:
```
from sagemaker.inputs import TrainingInput
s3_input_path = 's3://sudokudata/sudoku_sampled_1M.csv'  # your sampled file
train_input = TrainingInput(s3_input_path, content_type='csv')
```

In [ ]:
```
estimator.fit({'training': train_input})
```

```
[05/06/25 15:17:18] INFO     SageMaker Python SDK will collect telemetry t
                             understand our user's needs, diagnose issues,
                             additional features.
                             To opt out of telemetry, please disable via T
                             parameter in SDK defaults config. For more in
                             to
                             https://sagemaker.readthedocs.io/en/stable/ov
                             guring-and-using-defaults-with-the-sagemaker-
                    INFO     image_uri is not presented, retrieving image_
                             instance_type, framework etc.
[05/06/25 15:18:24] INFO     image_uri is not presented, retrieving image_
                             instance_type, framework etc.
                    INFO     Creating training-job with name:
                             pytorch-training-2025-05-06-15-17-18-605
```

```
2025-05-06 15:18:28 Starting - Starting the training job...
2025-05-06 15:18:42 Starting - Preparing the instances for training...
2025-05-06 15:19:14 Downloading - Downloading input data...
2025-05-06 15:19:44 Downloading - Downloading the training imag
e..................
2025-05-06 15:22:49 Training - Training image download completed. Training in
progress..bash: cannot set terminal process group (-1): Inappropriate ioctl fo
r device
bash: no job control in this shell
/opt/conda/lib/python3.9/site-packages/paramiko/pkey.py:100: CryptographyDepre
cationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.cipher
s.algorithms.TripleDES and will be removed from this module in 48.0.0.
  "cipher": algorithms.TripleDES,
/opt/conda/lib/python3.9/site-packages/paramiko/transport.py:259: Cryptography
DeprecationWarning: TripleDES has been moved to cryptography.hazmat.decrepit.c
iphers.algorithms.TripleDES and will be removed from this module in 48.0.0.
  "class": algorithms.TripleDES,
2025-05-06 15:23:02,317 sagemaker-training-toolkit INFO     Imported framework
sagemaker_pytorch_container.training
2025-05-06 15:23:02,340 sagemaker-training-toolkit INFO     No Neurons detecte
d (normal if no neurons installed)
2025-05-06 15:23:02,354 sagemaker_pytorch_container.training INFO     Block un
til all host DNS lookups succeed.
2025-05-06 15:23:02,358 sagemaker_pytorch_container.training INFO     Invoking
user training script.
2025-05-06 15:23:05,371 sagemaker-training-toolkit INFO     No Neurons detecte
d (normal if no neurons installed)
2025-05-06 15:23:05,425 sagemaker-training-toolkit INFO     No Neurons detecte
d (normal if no neurons installed)
2025-05-06 15:23:05,478 sagemaker-training-toolkit INFO     No Neurons detecte
d (normal if no neurons installed)
2025-05-06 15:23:05,500 sagemaker-training-toolkit INFO     Invoking user scri
pt
Training Env:
{
    "additional_framework_parameters": {},
    "channel_input_dirs": {
        "training": "/opt/ml/input/data/training"
    },
    "current_host": "algo-1",
    "current_instance_group": "homogeneousCluster",
    "current_instance_group_hosts": [
        "algo-1"
    ],
    "current_instance_type": "ml.g4dn.xlarge",
    "distribution_hosts": [],
    "distribution_instance_groups": [],
    "framework_module": "sagemaker_pytorch_container.training:main",
    "hosts": [
        "algo-1"
    ],
    "hyperparameters": {
        "batch-size": 64,
        "epochs": 5,
        "lr": 0.001
    },
    "input_config_dir": "/opt/ml/input/config",
    "input_data_config": {
        "training": {
            "ContentType": "csv",
            "TrainingInputMode": "File",
            "S3DistributionType": "FullyReplicated",
            "RecordWrapperType": "None"
        }
    },
    "input_dir": "/opt/ml/input",
    "instance_groups": [
        "homogeneousCluster"
    ],
    "instance_groups_dict": {
        "homogeneousCluster": {
            "instance_group_name": "homogeneousCluster",
            "instance_type": "ml.g4dn.xlarge",
            "hosts": [
                "algo-1"
            ]
        }
    },
    "is_hetero": false,
    "is_master": true,
    "is_modelparallel_enabled": null,
    "is_smddpmprun_installed": true,
    "is_smddprun_installed": true,
    "job_name": "pytorch-training-2025-05-06-15-17-18-605",
    "log_level": 20,
    "master_hostname": "algo-1",
    "model_dir": "/opt/ml/model",
    "module_dir": "s3://sagemaker-us-east-1-971422672957/pytorch-training-2025
-05-06-15-17-18-605/source/sourcedir.tar.gz",
    "module_name": "train",
    "network_interface_name": "eth0",
    "num_cpus": 4,
    "num_gpus": 1,
    "num_neurons": 0,
    "output_data_dir": "/opt/ml/output/data",
    "output_dir": "/opt/ml/output",
    "output_intermediate_dir": "/opt/ml/output/intermediate",
    "resource_config": {
        "current_host": "algo-1",
        "current_instance_type": "ml.g4dn.xlarge",
        "current_group_name": "homogeneousCluster",
        "hosts": [
            "algo-1"
        ],
        "instance_groups": [
            {
                "instance_group_name": "homogeneousCluster",
                "instance_type": "ml.g4dn.xlarge",
                "hosts": [
                    "algo-1"
                ]
            }
        ],
        "network_interface_name": "eth0"
    },
    "user_entry_point": "train.py"
}
Environment variables:
SM_HOSTS=["algo-1"]
SM_NETWORK_INTERFACE_NAME=eth0
```

```
SM_HPS={"batch-size":64,"epochs":5,"lr":0.001}
SM_USER_ENTRY_POINT=train.py
SM_FRAMEWORK_PARAMS={}
SM_RESOURCE_CONFIG={"current_group_name":"homogeneousCluster","current_hos
t":"algo-1","current_instance_type":"ml.g4dn.xlarge","hosts":["algo-1"],"insta
nce_groups":[{"hosts":["algo-1"],"instance_group_name":"homogeneousCluster","i
nstance_type":"ml.g4dn.xlarge"}],"network_interface_name":"eth0"}
SM_INPUT_DATA_CONFIG={"training":{"ContentType":"csv","RecordWrapperType":"Non
e","S3DistributionType":"FullyReplicated","TrainingInputMode":"File"}}
SM_OUTPUT_DATA_DIR=/opt/ml/output/data
SM_CHANNELS=["training"]
SM_CURRENT_HOST=algo-1
SM_CURRENT_INSTANCE_TYPE=ml.g4dn.xlarge
SM_CURRENT_INSTANCE_GROUP=homogeneousCluster
SM_CURRENT_INSTANCE_GROUP_HOSTS=["algo-1"]
SM_INSTANCE_GROUPS=["homogeneousCluster"]
SM_INSTANCE_GROUPS_DICT={"homogeneousCluster":{"hosts":["algo-1"],"instance_gr
oup_name":"homogeneousCluster","instance_type":"ml.g4dn.xlarge"}}
SM_DISTRIBUTION_INSTANCE_GROUPS=[]
SM_IS_HETERO=false
SM_MODULE_NAME=train
SM_LOG_LEVEL=20
SM_FRAMEWORK_MODULE=sagemaker_pytorch_container.training:main
SM_INPUT_DIR=/opt/ml/input
SM_INPUT_CONFIG_DIR=/opt/ml/input/config
SM_OUTPUT_DIR=/opt/ml/output
SM_NUM_CPUS=4
SM_NUM_GPUS=1
SM_NUM_NEURONS=0
SM_MODEL_DIR=/opt/ml/model
SM_MODULE_DIR=s3://sagemaker-us-east-1-971422672957/pytorch-training-2025-05-0
6-15-17-18-605/source/sourcedir.tar.gz
SM_TRAINING_ENV={"additional_framework_parameters":{},"channel_input_dirs":{"t
raining":"/opt/ml/input/data/training"},"current_host":"algo-1","current_insta
nce_group":"homogeneousCluster","current_instance_group_hosts":["algo-1"],"cur
rent_instance_type":"ml.g4dn.xlarge","distribution_hosts":[],"distribution_ins
tance_groups":[],"framework_module":"sagemaker_pytorch_container.training:mai
n","hosts":["algo-1"],"hyperparameters":{"batch-size":64,"epochs":5,"lr":0.00
1},"input_config_dir":"/opt/ml/input/config","input_data_config":{"training":
{"ContentType":"csv","RecordWrapperType":"None","S3DistributionType":"FullyRep
licated","TrainingInputMode":"File"}},"input_dir":"/opt/ml/input","instance_gr
oups":["homogeneousCluster"],"instance_groups_dict":{"homogeneousCluster":{"ho
sts":["algo-1"],"instance_group_name":"homogeneousCluster","instance_type":"m
l.g4dn.xlarge"}},"is_hetero":false,"is_master":true,"is_modelparallel_enable
d":null,"is_smddpmprun_installed":true,"is_smddprun_installed":true,"job_nam
e":"pytorch-training-2025-05-06-15-17-18-605","log_level":20,"master_hostnam
e":"algo-1","model_dir":"/opt/ml/model","module_dir":"s3://sagemaker-us-east-1
-971422672957/pytorch-training-2025-05-06-15-17-18-605/source/sourcedir.tar.g
z","module_name":"train","network_interface_name":"eth0","num_cpus":4,"num_gpu
s":1,"num_neurons":0,"output_data_dir":"/opt/ml/output/data","output_dir":"/op
t/ml/output","output_intermediate_dir":"/opt/ml/output/intermediate","resource
_config":{"current_group_name":"homogeneousCluster","current_host":"algo-1","c
urrent_instance_type":"ml.g4dn.xlarge","hosts":["algo-1"],"instance_groups":
[{"hosts":["algo-1"],"instance_group_name":"homogeneousCluster","instance_typ
e":"ml.g4dn.xlarge"}],"network_interface_name":"eth0"},"user_entry_point":"tra
in.py"}
SM_USER_ARGS=["--batch-size","64","--epochs","5","--lr","0.001"]
SM_OUTPUT_INTERMEDIATE_DIR=/opt/ml/output/intermediate
SM_CHANNEL_TRAINING=/opt/ml/input/data/training
SM_HP_BATCH-SIZE=64
```

```
SM_HP_EPOCHS=5
SM_HP_LR=0.001
PYTHONPATH=/opt/ml/code:/opt/conda/bin:/opt/conda/lib/python39.zip:/opt/conda/
lib/python3.9:/opt/conda/lib/python3.9/lib-dynload:/opt/conda/lib/python3.9/si
te-packages
Invoking script with the following command:
/opt/conda/bin/python3.9 train.py --batch-size 64 --epochs 5 --lr 0.001
2025-05-06 15:23:05,540 sagemaker-training-toolkit INFO     Exceptions not imp
orted for SageMaker TF as Tensorflow is not installed.
Loaded dataset shape: (1000000, 2)
Train: (900000, 2), Validation: (100000, 2)
=== Epoch 1/5 ===
/opt/conda/lib/python3.9/site-packages/numpy/core/fromnumeric.py:57: FutureWar
ning: 'DataFrame.swapaxes' is deprecated and will be removed in a future versi
on. Please use 'DataFrame.transpose' instead.
  return bound(*args, **kwds)
Processing chunk 1...
[2025-05-06 15:23:16.544 algo-1:65 INFO utils.py:28] RULE_JOB_STOP_SIGNAL_FILE
NAME: None
[2025-05-06 15:23:16.685 algo-1:65 INFO profiler_config_parser.py:111] User ha
s disabled profiler.
[2025-05-06 15:23:16.685 algo-1:65 INFO json_config.py:92] Creating hook from
json_config at /opt/ml/input/config/debughookconfig.json.
[2025-05-06 15:23:16.686 algo-1:65 INFO hook.py:206] tensorboard_dir has not b
een set for the hook. SMDebug will not be exporting tensorboard summaries.
[2025-05-06 15:23:16.686 algo-1:65 INFO hook.py:259] Saving to /opt/ml/output/
tensors
[2025-05-06 15:23:16.686 algo-1:65 INFO state_store.py:77] The checkpoint conf
ig file /opt/ml/input/config/checkpointconfig.json does not exist.
Processing chunk 2...
Processing chunk 3...
Processing chunk 4...
Processing chunk 5...
Processing chunk 6...
Processing chunk 7...
Processing chunk 8...
Processing chunk 9...
Processing chunk 10...
Processing chunk 11...
Processing chunk 12...
Processing chunk 13...
Processing chunk 14...
Processing chunk 15...
Processing chunk 16...
Processing chunk 17...
Processing chunk 18...
Processing chunk 19...
Processing chunk 20...
Processing chunk 21...
Processing chunk 22...
Processing chunk 23...
Processing chunk 24...
Processing chunk 25...
Processing chunk 26...
Processing chunk 27...
Processing chunk 28...
Processing chunk 29...
Processing chunk 30...
Processing chunk 31...
Processing chunk 32...
Processing chunk 33...
Processing chunk 34...
Processing chunk 35...
Processing chunk 36...
Processing chunk 37...
Processing chunk 38...
Processing chunk 39...
Processing chunk 40...
Processing chunk 41...
Processing chunk 42...
Processing chunk 43...
Processing chunk 44...
Processing chunk 45...
✅ Epoch 1 Training Accuracy: 73.48%
🧪 Epoch 1 Validation Accuracy: 80.76%
Saved checkpoint: /opt/ml/model/model_epoch0.pth
=== Epoch 2/5 ===
/opt/conda/lib/python3.9/site-packages/numpy/core/fromnumeric.py:57: FutureWar
ning: 'DataFrame.swapaxes' is deprecated and will be removed in a future versi
on. Please use 'DataFrame.transpose' instead.
  return bound(*args, **kwds)
Processing chunk 1...
Processing chunk 2...
Processing chunk 3...
Processing chunk 4...
Processing chunk 5...
Processing chunk 6...
Processing chunk 7...
Processing chunk 8...
Processing chunk 9...
Processing chunk 10...
Processing chunk 11...
Processing chunk 12...
Processing chunk 13...
Processing chunk 14...
Processing chunk 15...
Processing chunk 16...
Processing chunk 17...
Processing chunk 18...
Processing chunk 19...
Processing chunk 20...
Processing chunk 21...
Processing chunk 22...
Processing chunk 23...
Processing chunk 24...
Processing chunk 25...
Processing chunk 26...
Processing chunk 27...
Processing chunk 28...
Processing chunk 29...
Processing chunk 30...
Processing chunk 31...
Processing chunk 32...
Processing chunk 33...
Processing chunk 34...
Processing chunk 35...
Processing chunk 36...
Processing chunk 37...
Processing chunk 38...
Processing chunk 39...
Processing chunk 40...
Processing chunk 41...
Processing chunk 42...
Processing chunk 43...
Processing chunk 44...
Processing chunk 45...
✅ Epoch 2 Training Accuracy: 83.94%
🧪 Epoch 2 Validation Accuracy: 83.49%
Saved checkpoint: /opt/ml/model/model_epoch1.pth
=== Epoch 3/5 ===
/opt/conda/lib/python3.9/site-packages/numpy/core/fromnumeric.py:57: FutureWar
ning: 'DataFrame.swapaxes' is deprecated and will be removed in a future versi
on. Please use 'DataFrame.transpose' instead.
  return bound(*args, **kwds)
Processing chunk 1...
Processing chunk 2...
Processing chunk 3...
Processing chunk 4...
Processing chunk 5...
Processing chunk 6...
Processing chunk 7...
Processing chunk 8...
Processing chunk 9...
Processing chunk 10...
Processing chunk 11...
Processing chunk 12...
Processing chunk 13...
Processing chunk 14...
Processing chunk 15...
Processing chunk 16...
Processing chunk 17...
Processing chunk 18...
Processing chunk 19...
Processing chunk 20...
Processing chunk 21...
Processing chunk 22...
Processing chunk 23...
Processing chunk 24...
Processing chunk 25...
Processing chunk 26...
Processing chunk 27...
Processing chunk 28...
Processing chunk 29...
Processing chunk 30...
Processing chunk 31...
Processing chunk 32...
Processing chunk 33...
Processing chunk 34...
Processing chunk 35...
Processing chunk 36...
Processing chunk 37...
Processing chunk 38...
Processing chunk 39...
Processing chunk 40...
Processing chunk 41...
Processing chunk 42...
Processing chunk 43...
Processing chunk 44...
Processing chunk 45...
✅ Epoch 3 Training Accuracy: 86.27%
```

```
🧪 Epoch 3 Validation Accuracy: 84.98%
Saved checkpoint: /opt/ml/model/model_epoch2.pth
=== Epoch 4/5 ===
/opt/conda/lib/python3.9/site-packages/numpy/core/fromnumeric.py:57: FutureWar
ning: 'DataFrame.swapaxes' is deprecated and will be removed in a future versi
on. Please use 'DataFrame.transpose' instead.
  return bound(*args, **kwds)
Processing chunk 1...
Processing chunk 2...
Processing chunk 3...
Processing chunk 33...
Processing chunk 34...
Processing chunk 35...
Processing chunk 36...
Processing chunk 37...
Processing chunk 38...
Processing chunk 39...
Processing chunk 40...
Processing chunk 41...
Processing chunk 42...
Processing chunk 43...
Processing chunk 44...
Processing chunk 45...
✅ Epoch 4 Training Accuracy: 88.14%
🧪 Epoch 4 Validation Accuracy: 86.13%
Saved checkpoint: /opt/ml/model/model_epoch3.pth
=== Epoch 5/5 ===
/opt/conda/lib/python3.9/site-packages/numpy/core/fromnumeric.py:57: FutureWar
ning: 'DataFrame.swapaxes' is deprecated and will be removed in a future versi
on. Please use 'DataFrame.transpose' instead.
  return bound(*args, **kwds)
Processing chunk 1...
Processing chunk 2...
Processing chunk 3...
Processing chunk 4...
Processing chunk 5...
Processing chunk 6...
Processing chunk 7...
Processing chunk 8...
Processing chunk 9...
Processing chunk 10...
Processing chunk 11...
Processing chunk 12...
Processing chunk 13...
Processing chunk 14...
Processing chunk 15...
Processing chunk 16...
Processing chunk 17...
Processing chunk 18...
Processing chunk 19...
Processing chunk 20...
Processing chunk 21...
Processing chunk 22...
Processing chunk 23...
Processing chunk 24...
Processing chunk 25...
Processing chunk 26...
Processing chunk 27...
Processing chunk 28...
Processing chunk 29...
Processing chunk 30...
Processing chunk 31...
Processing chunk 32...
Processing chunk 33...
Processing chunk 34...
Processing chunk 35...
Processing chunk 36...
Processing chunk 37...
Processing chunk 38...
Processing chunk 39...
Processing chunk 40...
Processing chunk 41...
Processing chunk 42...
Processing chunk 43...
Processing chunk 44...
Processing chunk 45...
✅ Epoch 5 Training Accuracy: 89.26%
🧪 Epoch 5 Validation Accuracy: 86.85%
Saved checkpoint: /opt/ml/model/model_epoch4.pth
Final model saved to model.pth
2025-05-06 19:44:18,239 sagemaker-training-toolkit INFO     Waiting for the pr
ocess to finish and give a return code.
2025-05-06 19:44:18,239 sagemaker-training-toolkit INFO     Done waiting for a
return code. Received 0 from exiting process.
2025-05-06 19:44:18,240 sagemaker-training-toolkit INFO     Reporting training
SUCCESS

2025-05-06 19:44:22 Uploading - Uploading generated training model
2025-05-06 19:44:50 Completed - Training job completed
Training seconds: 15935
Billable seconds: 15935
```

In [5]:
```
import tarfile
bucket = 'sagemaker-us-east-1-971422672957'
key = 'sudoku-model-output/pytorch-training-2025-05-06-15-17-18-605/output/mode
local_tar_path = '/tmp/model.tar.gz'
```

In [6]:
```
s3 = boto3.client('s3')
s3.download_file(bucket, key, local_tar_path)
```

In [7]:
```
extract_dir = '/tmp/model'
os.makedirs(extract_dir, exist_ok=True)
with tarfile.open(local_tar_path, 'r:gz') as tar:
    tar.extractall(path=extract_dir)

model_path = os.path.join(extract_dir, 'model.pth')
```

In [8]:
```
import torch
import torch.nn as nn

class SudokuSolverCNN(nn.Module):
    def __init__(self, num_layers=16):
        super(SudokuSolverCNN, self).__init__()
        self.layers = nn.ModuleList()
        self.layers.append(nn.Conv2d(1, 512, kernel_size=3, padding=1))
        self.layers.append(nn.BatchNorm2d(512))
        self.layers.append(nn.ReLU())
        for _ in range(num_layers - 2):
            self.layers.append(nn.Conv2d(512, 512, kernel_size=3, padding=1))
            self.layers.append(nn.BatchNorm2d(512))
            self.layers.append(nn.ReLU())
        self.final_conv = nn.Conv2d(512, 9, kernel_size=1)

    def forward(self, x):
        for layer in self.layers:
            x = layer(x)
        return self.final_conv(x)

model = SudokuSolverCNN(num_layers=16)
model.load_state_dict(torch.load(model_path, map_location='cpu'))
model.eval()
```

```
/tmp/ipykernel_1270/1608795755.py:23: FutureWarning: You are using `torch.load
` with `weights_only=False` (the current default value), which uses the defaul
t pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See https://github.com/py
torch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a f
uture 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 e
xplicitly 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(model_path, map_location='cpu'))
```

Out[8]:
```
SudokuSolverCNN(
  (layers): ModuleList(
    (0): Conv2d(1, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
stats=True)
    (2): ReLU()
    (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
stats=True)
    (5): ReLU()
    (6): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_
stats=True)
    (8): ReLU()
    (9): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (11): ReLU()
    (12): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (14): ReLU()
    (15): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (16): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (17): ReLU()
    (18): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (19): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (20): ReLU()
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (23): ReLU()
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (26): ReLU()
    (27): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (28): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (29): ReLU()
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (32): ReLU()
    (33): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (34): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (35): ReLU()
    (36): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (37): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (38): ReLU()
    (39): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (40): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (41): ReLU()
    (42): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (43): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running
_stats=True)
    (44): ReLU()
  )
  (final_conv): Conv2d(512, 9, kernel_size=(1, 1), stride=(1, 1))
)
```

In [15]:
```
import numpy as np
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder

# Load test data from S3
test_df = pd.read_csv('s3://sudokudata/sudoku_test_data.csv')

# Preprocess test data (use same logic as training)
puzzles = np.array([list(p) for p in test_df['puzzle']], dtype=np.int8).reshape
solutions = np.array([list(s) for s in test_df['solution']], dtype=np.int8).res
puzzles_flat = puzzles.reshape(puzzles.shape[0], -1)
solutions_flat = solutions.reshape(-1, 1)

# Ideally, use the scaler/encoder from training, but if not available, fit on 
scaler = MinMaxScaler(feature_range=(0, 1)).fit(puzzles_flat)
encoder = OneHotEncoder(categories=[range(1, 10)], sparse_output=False).fit(so

puzzles_scaled = scaler.transform(puzzles_flat).reshape(-1, 9, 9)
solutions_encoded = encoder.transform(solutions_flat).reshape(-1, 9, 9, 9)

X_test = torch.tensor(puzzles_scaled, dtype=torch.float32).unsqueeze(1)
y_test = torch.tensor(solutions_encoded, dtype=torch.float32).permute(0, 3, 1,
```

In [11]:
```
print(test_df)
```

```
                                                puzzle  \
0    9004103760302074904780001020050090608695002077...
1    1560207008003675210025016800340029672070504106...
2    0830740505018037009700008236900002101405800760...
3    0950301046028010570105240690801070355260487001...
4    7063015005100684074900071008401026052604739189...
..                                                 ...
995  0260850033946128055004000000602004977159406800...
996  0186900002547009603901207850305408001023005474...
997  7480300520026083070302040092594001700670294304...
998  2006094750903200800068473298050709320205867044...
999  0869100207245389101594600374070003022057940016...

                                              solution
0    9524183766312574984789631522157498638695312477...
1    1564287398493675213725916845341829672976534186...
2    2839746515618237949746158236983472151425893763...
3    8957361246428913573175248699841672355263487911...
4    7263415895139684274982571638471926352654739189...
..                                                 ...
995  1267859433946128755874391268632514977159436829...
996  7186954322547389613961247859375428161823695474...
997  7489316529126583476352748192594831761675294384...
998  2386194757943256815168473298654719323295867144...
999  3869175247245389161594628374976813522357946816...

[1000 rows x 2 columns]
```

In [12]:
```
batch_size = 256
num_samples = X_test.shape[0]
```

```
correct = 0
total = 0

with torch.no_grad():
    for i in range(0, num_samples, batch_size):
        inputs = X_test[i:i+batch_size]
        labels = y_test[i:i+batch_size]
        outputs = model(inputs)
        _, predicted = torch.max(outputs, 1)
        _, labels_max = torch.max(labels, 1)
        correct += (predicted == labels_max).sum().item()
        total += labels_max.numel()

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

```
Test Accuracy: 99.10%
```

Although the test accuracy is 99%, it might not be completely reflective of the true performance of the model. Therefore, we will download a new dataset completely to sample 2,000 entries from.

In [14]:
```
different_data = pd.read_csv('new_sudoku_test.csv')
different_data.describe()
```

Out[14]:

	quizzes	
count	2000	
unique	2000	
top	9000075006010503000809010203600098040400200100...	924387561671254389583961427
freq	1	

In [16]:
```
different_data = different_data.rename(columns={'quizzes': 'puzzle', 'solution
different_data.describe()
```

Out[16]:

	puzzle	
count	2000	
unique	2000	
top	9000075006010503000809010203600098040400200100...	924387561671254389583961427
freq	1	

In [17]:
```
# Preprocess test data (use same logic as training)
puzzles = np.array([list(p) for p in different_data['puzzle']], dtype=np.int8)
solutions = np.array([list(s) for s in different_data['solution']], dtype=np.i
puzzles_flat = puzzles.reshape(puzzles.shape[0], -1)
solutions_flat = solutions.reshape(-1, 1)

# Ideally, use the scaler/encoder from training, but if not available, fit on
scaler = MinMaxScaler(feature_range=(0, 1)).fit(puzzles_flat)
encoder = OneHotEncoder(categories=[range(1, 10)], sparse_output=False).fit(so

puzzles_scaled = scaler.transform(puzzles_flat).reshape(-1, 9, 9)
```

```
solutions_encoded = encoder.transform(solutions_flat).reshape(-1, 9, 9, 9)

X_test = torch.tensor(puzzles_scaled, dtype=torch.float32).unsqueeze(1)
y_test = torch.tensor(solutions_encoded, dtype=torch.float32).permute(0, 3, 1,
```

In [18]:

```
batch_size = 256
num_samples = X_test.shape[0]
correct = 0
total = 0

with torch.no_grad():
    for i in range(0, num_samples, batch_size):
        inputs = X_test[i:i+batch_size]
        labels = y_test[i:i+batch_size]
        outputs = model(inputs)
        _, predicted = torch.max(outputs, 1)
        _, labels_max = torch.max(labels, 1)
        correct += (predicted == labels_max).sum().item()
        total += labels_max.numel()

accuracy = 100 * correct / total
print(f"Test Accuracy for test data from different data: {accuracy:.2f}%")
```

```
Test Accuracy for test data from different data: 91.31%
```

In [ ]: